МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова праця
на правах рукопису

ЛЕМЕЩУК ОЛЕКСАНДР ІГОРОВИЧ

УДК [004.4/.49:378.014.6-047.64]

ДИСЕРТАЦІЯ

РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ПРОЦЕСАМИ ЯКОСТІ ОСВІТИ УНІВЕРСИТЕТУ ЗА ДОПОМОГОЮ СЕРВІСІВ

121 Інженерія програмного забезпечення
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використані ідеї, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ О.І. Лемещук

Науковий керівник – Львов Михайло Сергійович,
доктор фізико-математичних наук, професор кафедри комп'ютерних наук та програмної інженерії Херсонського державного університету

Івано-Франківськ - Херсон – 2024

АНОТАЦІЯ

Лемешук О. І. Розробка системи управління процесами якості освіти університету за допомогою сервісів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення» – Державний університет України «Херсонський державний університет» МОН України; Державний університет України «Херсонський державний університет» МОН України; Херсон, 2024.

Дисертаційну роботу присвячено актуальній темі розробки та впровадження сучасного віртуального освітнього середовища як інструмента для ефективної організації освітнього процесу у закладі вищої освіти України. Дисертаційна робота зосереджується на функціональних можливостях учасників освітнього процесу. У якості основи для програмного комплексу який об'єднує інформаційні технології, методологічні та практичні аспекти керування та підтримки навчального процесу, різноманітних аспектах бізнес-процесів сучасного вищого навчального закладу, запропоновано використовувати досвід із розробки «систем керування взаємовідносинами клієнтів».

У вступі розкрито актуальність теми дослідження, історію розвитку напрацювань із розробки програмних засобів навчального призначення в закладах вищої освіти України, зокрема у Херсонському державному університеті. У досліджені систематизовано багаторічний досвід із розробки та впровадження віртуальних освітніх середовищ у закладах вищої освіти України та світу.

На основі ретельного аналізу функціональних вимог до програмних засобів, що входять до складу віртуальних освітніх середовищ, розроблених та впроваджених закладами вищої освіти України та світу сформовано функціональні та апаратні вимоги до нового віртуального освітнього середовища, яке спрямовано на рішення проблем організації

та вдосконалення освітнього процесу у сучасних закладах вищої освіти в Україні. Функціональні вимоги згруповані за категоріями та відповідають ролям користувачів у програмному комплексі.

Запропоновано методологію організації освітніх та адміністративних бізнес-процесів у закладах вищої освіти України, використання новітніх технологій, на прикладі «blockchain» з метою покращення процесів обміну професійними здобутками здобувачів вищої освіти із майбутніми роботодавцями.

У роботі систематизовано досвід науково-дослідних лабораторій Херсонського державного університету із розробки навчальних середовищ призначених для середньої та старшої шкіл України з математичних дисциплін, основ програмування; досвід із розробки віртуальних університетів та інтегрованих середовищ навчання для закладів вищої освіти.

Розроблено технічне завдання та програмні модулі віртуального освітнього середовища «KSU24». Запроваджено методологію та механізми внесення змін у програмний код, відповідно до мінливих вимог науково-педагогічних працівників та адміністрації закладу вищої освіти.

Обґрунтовано впровадження віртуального освітнього середовища «KSU24» для автоматизації бізнес процесів закладу вищої освіти України. Обґрунтовано реалізацію нового віртуального освітнього середовища у вигляді розподіленого інтернет застосунка, побудованого на основі шаблону проектування «модель - вид - представлення», із використанням сучасних каркасів побудови інтернет застосунків «Django» мови програмування Python для розробки серверної частини віртуального освітнього середовища, та «React» мови програмування JavaScript, для розробки клієнтської частини віртуального освітнього середовища.

Оцінено ефективність освітніх та адміністративних процесів закладу вищої України, для автоматизації бізнес процесів яких залучено

віртуальне освітнє середовище у порівнянні із неавтоматизованими бізнес-процесами закладів вищої освіти.

Визначено функціональні вимоги, апаратні вимоги, та категорії користувачів для типового віртуального освітнього середовища закладів вищої освіти України.

Удосконалено та автоматизовано супровідний документообіг типових бізнес-процесів закладу вищої освіти України. Додано сучасну та гнучку систему для генерації звітної документації будь-якого рівня складності за будь-якими ролями та параметрами, що реалізовані у системі використовуючи програмну бібліотеку «Pandas».

Проведено ретельний аналіз типових програмних засобів впроваджених в освітні процеси типових закладів вищої освіти України. На його основі побудовано вимоги до більшості програмних модулів, що входять до складу віртуального освітнього середовища, та додано механізми інтеграції з іншими програмними засобами навчального призначення.

Автором виокремлено архітектурні та програмні рішення для ефективної побудови складових частин віртуального освітнього середовища для закладів вищої освіти України.

Практичне значення одержаних результатів полягає у механізмах покращення можливостей адаптації закладів вищої освіти України до мінливих умов сучасного світу, які використано при переміщенні Херсонського державного університету до іншого міста через військову агресію.

Ключові слова: віртуальне освітнє середовище, бізнес-процеси закладу вищої освіти, автоматизація, освітній процес, звітній документообіг.

ANNOTATION

Lemeshchuk O. Development of the University Quality Management System by means of Services. – Qualifying scientific work on manuscript rights.

Dissertation for obtaining the scientific degree a Doctor of Philosophy in the specialty 121 «Program engineering» – Ukrainian state university «Kherson State University» MES of Ukraine; Ukrainian State University «Kherson State University » MES of Ukraine; Kherson, 2024.

The dissertation is dedicated to the relevant theme of development and implementation of a modern virtual learning environment as an instrument for the effective organization of educational processes in Ukrainian higher education institutions. As a basis for a software complex that combines information technologies, methodological, and practical aspects of management and support of the educational process, as well as various aspects of business processes of a modern higher education institution, it was first proposed to use experience in the development of "customer relationship management systems".

The introduction reveals the relevance of the research, the history of educational software development in Ukrainian higher education institutions, and Kherson State University. The Ukrainian virtual learning environments development experience is also revealed.

Based on the detailed functional requirements analysis of the software that is a part of virtual learning environments developed and maintained by Ukrainian universities, the research formulates the functional and software requirements for building a new virtual learning environment dedicated to improving the educational processes of modern Ukrainian higher education institutions. Based on this, the requirements for most of the software modules included in the virtual educational environment were built, and integration mechanisms with other educational software tools were added.

The methodology for the organization of educational and administrative business processes in Ukrainian higher education institutions that uses new kinds of technologies similar to «blockchain» is proposed, in order to improve the processes of experience exchange between graduate students and future stakeholders.

The experience of Kherson State University's scientific laboratories in developing educational environments for high schools, focused on mathematical and programming disciplines, is systematized.

The requirements specification for the virtual learning environment «KSU24» is developed.

The methodology and technology for making changes to the source code according to the ever-changing requirements of teachers and administration of the higher education institution are detailed.

The implementation of the virtual educational environment «KSU24» for the automation of business processes of higher education institutions in Ukraine is substantiated.

The effectiveness of the educational and administrative processes of higher education institutions in Ukraine, for which a virtual educational environment was involved, was evaluated in comparison with non-automated business processes of higher education institutions.

The accompanying document flow of typical business processes of a higher education institution in Ukraine is improved and automated. A modern and flexible system for generating reporting documentation of any level of complexity for any roles and parameters, implemented in the system using the "Pandas" software library, was also added.

A detailed analysis of the typical software implemented in the educational process of higher education institutions in Ukraine was performed.

The author singles out architectural and software solutions for the effective construction of a virtual learning environment for higher education institutions in Ukraine.

The practical significance of the obtained results lies in the mechanisms for improving the adaptation capabilities of higher education institutions in Ukraine to the changing conditions of the modern world, which were used when Kherson State University was moved to another city due to military aggression.

Key words: virtual learning environment, higher education institution business processes automation, educational process, reporting document flow.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API – Applicated Programming Interface (прикладний програмний інтерфейс)

CDP – Customer data platform (Платформи клієнтських даних)

CMS – Content Management System (системи керування вмістом)

CRM – Customer relationship management (Системи управління відносинами)

EDM – Entity Data Model (інтелектуальний аналіз даних навчального процесу)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

ITS – Intelligent Tutoring System (Інтелектуальна навчальна система)

LMS – Learning Management System (системи керування навчанням)

LTI – Learning Tools Interoperability (взаємодія інструментів навчання)

MVC – Model-View-Controller (Модель-представлення-контролер)

ORM – Object–Relational Mapping (Об’єктно-реляційне відображення)

SCORM – Sharable Content Object Reference Model (Розподілена еталонна модель об’єкта контенту)

SIS – Student Information System (Здобувач вищої освітиська інформаційна система)

SPA – Single Page Application (односторінкова програма)

VLE – Virtual Learning Environment (віртуальне освітнє середовище)

ЗВО – заклад вищої освіти

ІАС – інформаційно-аналітична система

ІТ – інформаційні технології

ПК – персональний комп’ютер

ВСТУП

Актуальність теми. Дистанційна форма освіти в Україні впроваджується вже більше десяти років. Епідемія коронавірусної інфекції, та перехід більшої частини підприємств до віддаленої роботи, значно посприяли пришвидшенню відповідних процесів в освіті в усьому світі. Хоча дистанційне навчання має певну кількість недоліків у порівнянні із традиційним, серед яких насамперед виділяються потреби у відповідному апаратному та програмному забезпеченні з боку закладів освіти, необхідність набуття додаткових навичок роботи із інформаційно-комунікаційними технологіями з боку викладацького складу, його впровадження є актуальним та необхідним в реаліях сучасної освіти в Україні та світі, оскільки цей підхід дозволяє здобувачам освіти вирішувати питання, які пов’язані із віддаленістю та транспортом, виключати моменти затримки у приєднанні до навчальних занять та ефективніше залучати до навчального процесу науково-педагогічні працівників, експертів, або фахівців з відповідного фаху до освітнього процесу, які працюють віддалено [183]. На законодавчому рівні у теперішній час порядок організації та впровадження дистанційної освіті визначається «Положенням про дистанційну форму здобуття повної загальної середньої освіти», який затверджено наказом Міністерства освіти і науки України від 08.09.2020 № 1115.

У сучасному світі, у переважній більшості англомовних країн основною технологією організації дистанційного освітнього процесу є віртуальні освітні середовища — інтернет-платформи, що поєднують як головні аспекти підтримки навчального процесу, так і інструменти керування фінансовими та правовими аспектами у відповідних університетах.

В Україні серед найвідоміших інформатизованих систем організації та підтримки освітнього процесу виділяються наступні:

1. АСУ «ВНЗ». Автоматизована система керування ЗВО всіх

рівнів акредитації, що має модульну структуру, яка дозволяє закладам вищої освіти обирати найнеобхідніші саме ним функції та максимально ефективно інтегрувати функціональні можливості системи у власну інформаційну інфраструктуру [16].

2. Автоматизована система управління вищим навчальним закладом III– IV рівня акредитації. ТОВ «Юнітех+» 2019 [17].
3. Автоматизована система управління навчальним процесом АСК «ВНЗ» [13].
4. Система управління навчальним процесом для вищих навчальних закладів «Директива» [14].
5. Пакет програм «Деканат» [15].

Дослідження в напрямку розробки навчальних середовищ у Херсонському державному університеті велись на кафедрі інформатиці та в лабораторіях «Розробки та впровадження педагогічних програмних засобів», «Інтегрованих середовищ навчання», «Мультимедійних та Дистанційних Технологій» при Науково Дослідному Інституті Інформаційних технологій та відділі «Інформаційно обчислювальний центр». В цих відділах під керівництвом проф. Львова М. С. були розроблені наступні навчальні середовища:

1. Програмно-методичний комплекс “ТерМ VII” підтримки практичного вивчення математики [3].
2. Інтегроване середовище вивчення курсу “Аналітична геометрія, для вищих навчальних закладів» [7].
3. Програмний комплекс «Бібліотека електронних наочностей Алгебра 7-9 для старшої школи в Україні» [7-9].
4. Програмно-методичний комплекс «Геометрія, 8 клас» [2].
5. Програмно-педагогічний комплекс «Відеоінтерпретатор алгоритмів пошуку та сортування» [4].
6. Програмно-методичний комплекс «Алгебра 7 клас» [7].
7. Програмно-методичний комплекс «Алгебра 8 клас» [9].

Під керівництвом Співаковського О. В. розроблено наступні навчальні середовища:

1. Система керування навчальною та адміністративно-господарською частинами життя вищого навчального закладу – «Інформаційно Аналітична система» (скор. ІАС) [1].
2. Програмний засіб «Системи лінійних рівнянь» [5].
3. Програмно методичний комплекс «Світ лінійної алгебри» [12].

У зв’язку із все більш прискореним розвитком технологій у сучасному світі інформаційні технології та освіта ставали все більше взаємопов’язаними та у 2004 році співробітниками відділу «Мультимедійних та дистанційних технологій навчання» під керівництвом Кравцова Г.М. було розроблено систему дистанційного навчання «Херсонський віртуальний університет»[10], а окремі функціональні частини розроблених раніше систем були адаптовані для роботи у мережі інтернет у якості нових навчальних середовищ. Так, під керівництвом Співаковського О.В. та Круглика В.С. на основі «Світ лінійної алгебри»[12], було розроблено інтегроване середовище навчання «WebAlmir»[11], під керівництвом Співаковського О.В. та Осипової Н.В. було розроблено інтегроване середовище навчання «WebOAP» [6] та KSU-online.

Зв'язок роботи з науковими планами й темами. Робота проводиться в рамках науково дослідної роботи за темою «Виконання завдань перспективного плану розвитку наукового напряму «Технічні науки» в Херсонському державному університеті за період з 2021 по 2026 роки» (державний реєстраційний номер 0122U000045), що проводиться з метою «Створення організаційних, ресурсних умов, які забезпечують можливість участі у науковій діяльності молоді для вирішення питань побудови інсерційної моделі загальної програмної системи навчального призначення з використанням методів комп’ютерної алгебри».

Мета і завдання дослідження. Мета дослідження полягає в розробці, впровадженні, організації підтримки та інтеграції у поточні навчальні та бізнес-процеси Херсонського державного університету програмного комплексу «KSU24» .

Для досягнення цієї мети в дисертації було вирішено наступні **завдання**:

- аналіз сучасних підходів та інформаційних технологій для підтримки навчального процесу у закладах вищої освіти;
- характеристика сучасних підходів та інформаційних технологій для підтримки економічно-правових процесів у закладах вищої освіти;
- застосування методології побудови віртуальних навчальних середовищ для створення сучасного інформаційного середовища у закладах вищої освіти України;
- створення технічного завдання та тестового плану для розробки та впровадження віртуального навчального середовища у вищі навчальні заклади України, спираючись на досвід Херсонського державного університету;
- розроблення серверної частини та бази даних програмного засобу для підтримки навчальних та бізнес процесів університету;
- розроблення інтерфейсу, та клієнтської частини програмного засобу відповідно до сучасної методології створення користувацьких інтерфейсів — “Ant-design”;

Об’єкт дослідження — віртуальні освітні середовища і засоби їх розробки та впровадження у закладах вищої освіти України.

Предмет дослідження — віртуальне освітнє середовище та бізнес-процеси Херсонського державного університету на платформі «KSU24» . Складові частини, засоби організації, підтримки та аналізу результатів навчальних, фінансових, адміністративних та інших бізнес-процесів

університету.

Методи дослідження: Теоретичною основою дослідження стали положення теорій електронної освіти. Теоретична основа дисертаційної роботи сформована шляхом застосування системного, структурного підходів до побудови віртуальних освітніх середовищ. Методичну основу роботи склав комплекс таких загальнонаукових та спеціальних методів: аналізу ти синтезу, аналогії та моделювання, системного аналізу. Досягнення поставленої мети та вирішення завдань забезпечило використання наступних наукових підходів, загальнонаукових і спеціальних методів: монографічного методу, абстрактно-логічного, логічних узагальнень – для розвитку понятійнотермінологічного апарату та теоретичних узагальнень освітніх процесів у закладах вищої освіти України; методів діалектичного пізнання і формалізації, причинно-наслідкового зв'язку – для розроблення функціональних вимог щодо роботи та впровадження віртуального освітнього середовища закладу вищої освіти України; наукової абстракції, конкретизації, аналізу і синтезу – для формування ролей та їх категорій користувачів віртуального освітнього середовища для закладу вищої освіти України; методів кластерного аналізу – для розробки моделей формування, аналізу та обробки відповідних виразів даних для генерації звітної інформації на основі результатів академічної успішності здобувачів вищої освіти; еволюційного програмування – для реалізації моделей та шляхів розвитку віртуального освітнього середовища враховуючи мінливі вимоги до освітнього процесу в Україні.

Інформаційною базою дослідження стали: закони України, нормативно-правові акти Кабінету Міністрів України, офіційні дані Державної служби статистики України та Європейського статистичного бюро, Організації економічного співробітництва і розвитку, Агентства з розвитку інфраструктури фондового ринку України, дані фінансової та управлінської звітності підприємств, результати власних досліджень,

наукові публікації, Інтернет-ресурси.

Дослідження виконувались у рамках зареєстрованої теми «Теоретичні основи та практичні результати побудови сервіс-орієнтованої програмної системи управління ЗВО (на прикладі KSU24)».

Наукова новизна отриманих результатів. Наукова новизна й теоретична значущість дослідження полягає в тому, що: вперше спроектовано, реалізовано та впроваджено повноцінне віртуальне навчальне середовище, до складу якого входять програмні комплекси, які охоплюють повний спектр бізнес-процесів закладу вищої освіти України: від програмних систем підтримки організації, проведення та ведення усієї необхідної документації відносно навчального процесу університету із повним аналізом результатів навчання та рівня задоволеності здобувачів до систем керування адміністративними, господарськими, економічними та юридичними бізнес-процесами.

Практичне значення отриманих результатів. Реалізація та впровадження віртуальних навчальних середовищ у закладі вищої освіти України дозволяє системі освіти більш ефективно адаптуватись до сучасних реалій: епідемій COVID, воєнного стану завдяки застосування сучасних інформаційних технологій і методології дистанційного навчання.

Протягом останніх десяти років в Херсонському державному університеті розроблювалось та впроваджувалось віртуальне навчальне середовище, до складу якого входила велика кількість окремих сервісів, кожний з яких зосереджувався на одному конкретному бізнес-процесі. Програмний комплекс «KSU24» , описаний в роботі, об’єднує та аналізує результати роботи існуючих сервісів університету. Його впровадження призвело до суттєвої автоматизації та оптимізації навчальних, господарських та інших бізнес-процесів в університеті.

Одержані результати можуть бути використані в подальшому для передачі досвіду з розробки та впровадження віртуальних навчальних

середовищ іншим закладам вищої освіти України. Інтеграція між такими системами та обмін досвідом у майбутньому дозволять організувати та створити єдиний віртуальний навчальний простір, який відповідатиме сучасним освітнім вимогам як в Україні, так і в світі.

Особистий внесок здобувача. Дисертаційна робота є самостійно виконаним науковим дослідженням. Наукові результати, висновки та практичні розробки належать особисто автору. З наукових праць, опублікованих у співавторстві, в дисертації використано лише ті положення та ідеї, які є результатом власних досліджень здобувача. Особистий внесок у спільно опублікованих працях: у [183] – здобувачем обґрунтовано особливості застосування інноваційних інформаційних технологій управління закладом вищої освіти в умовах переміщення співробітників та інфраструктури закладу вищої освіти до іншого міста для забезпечення стійкості роботи закладу вищої освіти в кризові періоди; у [184] – здобувачем запропоновано варіант реалізація алгоритму пошуку найкоротших шляхів у графі із обмеженими ресурсам сучасними мовами програмування, та програмні засоби та бібліотеки для аналізу ефективності цього алгоритму; у [185] – здобувачем запропоновано підхід до оптимізації алгоритмів аналізу великих обсягів інформації; у [186] – здобувачем запропоновано новітні технологічні підходи до оптимізації процесів отримання зворотного зв'язку від здобувачів вищої освіти, та аналізу його результатів.

Апробація результатів дисертації. Основні положення, результати, висновки та пропозиції роботи обговорювали на наукових семінарах кафедри комп’ютерних наук та програмної інженерії факультету комп'ютерних наук, фізики та математики. Отримані результати здобувач доповідав й отримував схвалення на науково-практичних конференціях і семінарах всеукраїнського та міжнародного рівнів, зокрема на конференціях: «IX International scientific and practical conference SCIENCE AND PRACTICE: IMPLEMENTATION TO MODERN

SOCIETY 2021» 18-19 квітня, Манчестер, Велика Британія; XIV Міжнародної науково-практичної коференції «Інформаційні технології і автоматизація – 2021» 21-22 жовтня м.Одеса; «IntelITSIS'2023: 4th International Workshop on Intelligent Information Technologies and Systems of Information Security, Khmelnytskyi, Ukraine, 22–24 March 2023», «Information and Communication Technologies in Education, Research, and Industrial Applications 18th International Conference, ICTERI 2023. Ivano-Frankivsk, Ukraine».

Публікації. Основні результати дисертаційної роботи викладено в 6 працях. Серед них у виданнях України та зарубіжжя, що включені до міжнародних наукометричних баз, – 4. Крім того, 2 праці опубліковано в матеріалах міжнародних наукових конференцій. Опубліковані праці висвітлюють зміст дисертації.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, основної частини, що містить три розділи, висновків, списку використаних джерел, додатків. Повний обсяг дисертаційної роботи – 216 сторінки, із яких 182 сторінок – основний текст (таблиць – 21, рис. – 65, додатків - 6).

РОЗДІЛ 1
ТЕОРІЯ СУЧАСНИХ БІЗНЕС-ПРОЦЕСІВ УПРАВЛІННЯ ЗВО

1.1. Огляд літератури та програмного забезпечення з теми дослідження

Віртуальні освітні середовища та інші навчальні онлайн-платформи стали важливою частиною сучасної освіти та навчання, особливо через недавню пандемію. До найбільш поширених та потужних віртуальних освітніх середовищ, які спрямовані на створення гнучкого та інклюзивного навчального досвіду відносяться:

- платформа мікронавчання EdApp [21], яка підтримує віртуальні класи, обговорення, форуми та системи створення опитувань;
- онлайн-платформа для навчання LearnWorlds [22], платформа, спрямована на створення персоналізованого навчального досвіду для здобувач вищої освіти за допомогою “програвача курсів”, шаблонів, електронних книг і соціальної спільноти;
- освітнє програмне забезпечення на основі віртуальної реальності, яке зосереджується на організації навчання в авіаційній галузі V360E [23];
- комплексна система управління навчанням і віртуальне навчальне середовище, яке підтримує мультимодальне навчання, шляхом використання аудіо-, відео- або текстове спілкування в реальному часі — WizIQ [24];
- система керування освітніми організаціями наповнена такими функціями, як керування онлайн-заняттями, відстеження прогресу, онлайн-сертифікат і внутрішня соціальна мережа, яку учні можуть використовувати для спілкування один з одним і участі в групових обговореннях — Edvance360 [25];
- віртуальне освітнє середовище, яке дозволяє учням проходити заняття, відповідати на тести та опитування, надсилати домашні завдання, обмінюватися файлами та змагатися з іншими членами команди на будь-

якому пристрої — WooClap [26];

- веборієнтоване віртуальне освітнє середовище та система керування контентом, яке надає такі функції, як інтерактивна дошка, обмін миттєвими повідомленнями та можливість записувати заняття, що може бути особливо корисним для викладання мов та онлайн-репетиторства — Raven360 [27].

На сучасному етапі розвитку вищої освіти в Україні завдання з автоматизації та інформатизації вирішаються вищими навчальними закладами двома способами: або шляхом придбання готових програмних рішень для розв'язання та оптимізації конкретних окремих задач (зокрема задачі планування навчального навантаження), або шляхом розробки власних програмних систем керування навчальним процесом. Аналіз ринку програмного забезпечення в Україні показує, що більшість існуючих систем зосереджуються на розв'язанні однієї або невеликої кількості задач, які стосуються керування вищим навчальним закладом. Серед відомих на ринку України подібних систем виділяються наступні:

- АСУ «ВНЗ». Автоматизована система керування ЗВО всіх рівнів акредитації [16], яка складається з наступних основних компонент: «АС Приймальна комісія», «АС Деканат», «АС Студмістечко» та наступних додаткових: «АС Тест», «АС Конструктор звітів», «Менеджер резервних копій», «Веб розклад», «Веб деканат» та «Система контролю доступу»;
- автоматизована система управління вищим навчальним закладом III– IV рівня акредитації. ТОВ «Юнітех+» 2019 [17].
- автоматизована система управління навчальним процесом для вищих навчальних закладів усіх рівнів акредитації АСК «ВНЗ», розроблена у НДІ прикладних інформаційних технологій, яка є частиною інформаційно-виробничої системи «Освіта» [13];
- система управління навчальним процесом для вищих навчальних закладів «Директива», розроблена у ТОВ «Комп'ютерні інформаційні технології» [13];

- пакет програм «Деканат», розроблений ПП «Політек-СОФТ», до складу якого входить модуль «ПС Здобувач вищої освіти» [14].

Також у багатьох великих ЗВО України впроваджено та успішно використовуються власні розробки подібних систем:

– інформаційно-аналітична система керування вищим навчальним закладом «Університет» Херсонського державного університету[1];

– автоматизована база даних Центрального інституту післядипломної педагогічної освіти[18];

– засоби автоматизації управління навчальним закладом, що діють в НУ «Львівська політехніка» та ЛНУ імені Івана Франка;

– автоматизована інформаційна система «Електронний університет», створена у Хмельницькому національному університеті[19];

– комплексна система автоматизації управління навчальним процесом, розроблена й введена в експлуатацію у Львівському інституті банківської справи Університету банківської справи, м. Київ (ЛІБС УБС НБУ)[20].

1.2. Функціональні та нефункціональні вимоги сучасних інформаційних освітніх середовищ

Задача перенесення функціональності усіх існуючих інформаційних середовищ та програмних засобів Херсонського Державного університету в інтернет разом із повною інформатизацією бізнес процесів університету стала особливо актуальною під час пандемії, коли більшість людей була вимушена знаходитись в ізоляції, що також призвело до прискореного розвитку дистанційного навчання, онлайн курсів та засобів комунікації між здобувачам вищої освіти, іншими учасниками освітнього процесу, та рядових співробітників закладів освіти.

Ці фактори зумовили фундаментальні зміни в педагогічних процесах та інформаційних технологіях, спрямованих на їх підтримку [33]. Навчальний процес на усіх рівнях (від шкільного до закладів вищої

освіти) практично повністю перемістився із звичайних академічних аудиторій до віртуальних, долучатися до яких усі учасники навчального процесу можуть як із персональних стаціонарних комп'ютерів, так із мобільних пристроїв (планшетів, смартфонів та ін) [36].

Із сукупності цих факторів випливають функціональні вимоги до сучасного освітнього середовища «KSU24» , відповідно до груп ролей користувачів системи. В інформаційному середовищі «KSU24» , яке створено для підтримки усіх бізнес-процесів Херсонського Державного Університету (навчальних, адміністративних, юридичних, фінансових), виділяють більше 40 ролей користувачів з різними рівнями доступу [32]. Для цих ролей користувачів, згрупованих за відповідними категоріями, визначаються наступні **функціональні** вимоги :

- здобувачі вищої освіти ролі (абітурієнти, здобувачі вищої освіти, аспіранти, випускники):

- доступ до перегляду персоніфікованого розкладу здобувача вищої освіти;

- спілкування та співпраця - електронна пошта, повідомлення, чати, блоги та ін;

- доступ до перегляду персональних журналів академічних груп, залікових книжок, навчальних відомостей;

- доступ до завдань, тестів, контрольних робіт із відповідних дисциплін;

- викладацькі ролі — асистенти, науково-педагогічні працівники, доценти, професори;

- доступ до перегляду персоніфікованого розкладу викладача;

- спілкування та співпраця - електронна пошта, повідомлення, чати, блоги та ін;

- управління контентом — створення, редагування, зберігання та керування доступом до створення, редагування та використання навчальних матеріалів;

- доступ до перегляду персональних журналів академічних груп, залікових книжок, навчальних відомостей;
- доступ до створення академічних журналів, навчальних відомостей із відповідних дисциплін;

- ролі НПП — співробітники деканату, співробітники кафедр, диспетчера;
  - розробка та оформлення навчальних планів;
  - спілкування та співпраця - електронна пошта, повідомлення, чати, блоги та ін;
  - перегляд, створення та редагування навчального розкладу відповідного рівня (кафедри, факультету);
  - перегляд, перевірка навчальних журналів, відомостей, залікових книжок;
  - перегляд звітної інформації по відповідній структурній одиниці залежно від рівня доступу (наприклад, декан отримує інформацію про студентів-контрактників, які є боржниками) ;
- ролі служб контролю процесів — співробітники відділів якості освіти, навчального відділу, відділу забезпечення інформаційно-комунікаційної структури, бібліотеки, та інші;
  - спілкування та співпраця - електронна пошта, повідомлення, чати, блоги та ін;
  - затвердження навчальних журналів, відомостей, залікових книжок (навчальний відділ);
  - створення необхідної звітної документації відповідно до відділу;
- Адміністративні ролі — усі керівники відділів та системні адміністратори;

Нефункціональні вимоги інформаційної освітньої системи випливають із того, що така система є клієнт-серверним застосунком [31]:

- серверна частина вимагає апаратного забезпечення, відповідної

потужності:

- об’єм жорсткого диску не менше 500 ГБ для зберігання не тільки основної бази даних, а також системи що відповідає за збереження історії відвідувань, історій користувачів та «кєш»- бази даних для прискорення роботи системи;
- об’єм оперативної пам’яті не менше ніж 32 ГБ для забезпечення стабільної роботи сервісів
- якісне та безперервне підключення до мережі інтернет;

- Клієнтська частина для користування системою вимагає лише інтернет-браузера актуальної версії (Google Chrome 122.0.6261.128, Firefox 125.01, InternetExplorer 11, Safari 17.4.1);
- Інформаційна освітня система вимагає застосування сучасних стандартів, та форматів обміну інформацією (JSON редакції ECMA-404);
- Інформаційна освітня система забезпечує інструменти інтеграції із зовнішніми програмними системами підтримки навчального процесу, такими як:
  - системи керування навчанням (англ. Learning Management Systems);
  - системами керування фінансовою частиною навчального процесу;
  - системами забезпечення комунікації між здобувачам вищої освітии, викладачами, адміністрацією закладу вищої освіти;
- Інформаційна освітня система вимагає інструментів забезпечення безпеки даних, які використовуються в мережі інтернет:
  - Наборів стандартів та специфікацій для систем дистанційного навчання - SCORM (Sharable Content Object Reference Model). Останні версії формату SCORM було оформлено у вигляді Experience API

Ґрунтуючись на функціональних та нефункціональних вимогах

переходимо до обґрунтування підходу створення віртуальних освітніх середовищ.

1.3. Сучасний підхід до створення віртуальних освітніх середовищ

Моделі навчання значно змінилися за останні роки [30], завдяки сплеску дистанційного навчання в багатьох країнах (рис.1.1). Статистичні дані «Національного центру статистики освіти» США підкреслюють все зростаючу популярність дистанційного навчання та підкреслюють необхідність адаптації закладів освіти.

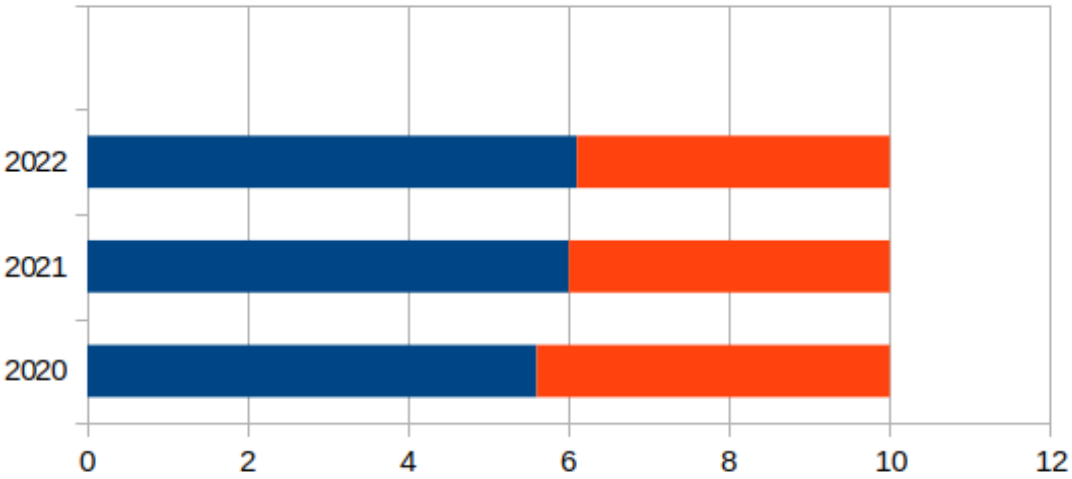


Рисунок 1.1. Поступове зростання відсотку залученості студентів у США

Для надання закладам освіти інструментів адаптації до нових тенденцій і забезпечення проведення якісних занять відповідно до вподобань здобувачів вищої освіти, віртуальні навчальні середовища мають вирішальне значення.

Для розробки платформи необхідно визначити наступні характеристики:

- формат пропонованого контенту;
- канали взаємодії;
- наявність можливостей налаштування та інструментів управління;
- існуючі рішення порівняно з потребами закладу освіти.

Новітні технологічні досягнення дозволяють спростити навчальний процес, процеси контролю результатів навчання на певних етапах, процеси, пов’язані із менеджментом, і як результат саму педагогічну модель в цілому. Також використання сучасних інформаційних технологій має велику перевагу перед традиційними підходами до навчання, оскільки більша частина учасників навчального процесу, а саме: діти, підлітки, вже дуже тісно інтегровані у сучасний інформаційний світ, ніж старше покоління.

Рисунок 1.2. Підсистеми які входять у склад сучасного віртуального освітнього середовища

Головна проблема управління освітнім процесом — це налагодження взаємодії між академічними установами та його основною аудиторією: здобувачами вищої освіти та батьками. Інформаційні технології надають інструменти оптимізації цієї взаємодії. Найбільш сучасним підходом у цій галузі є розробка та впровадження *віртуального освітнього середовища* (англ. Virtual Learning Environment) [32], яке складається із наступного набору підсистем (рис 1.2). *Віртуальне освітнє середовище* визначає простір, який не просто намагається фізично замінити проведення занять в академічних аудиторіях, а насамперед, збагатити його, використовуючи можливості сучасних потужних

програмних засобів та систем. У найбільш базовій формі функціональні вимоги до віртуального освітнього середовища обмежуються задачами розміщення навчальних курсів та документів, до яких мають доступ як здобувачі вищої освіти, так і науково-педагогічні працівники.

Це надає здобувачам вищої освіти мінімальний простір для взаємодії: отримання та проходження онлайн-курсів. Функціональні задачі науково-педагогічні працівників у цьому випадку: перевірка завдань від здобувач вищої освітиів, та виставлення відповідних оцінок [34].

Функціональні вимоги більшості освітніх установ, а особливо заклади вищої освіти, є набагато ширшими, тому вони використовують зовнішні програмні середовища, найчастіше за все – *системи керування навчанням* (англ. Learning management system) [35].

До сучасних віртуальних навчальних середовищ виділяють наступні функціональні вимоги (таблиця 1.1).

Таблиця 1.1

Основні функціональні вимоги в сучасному віртуальному освітньому середовищі

Функціонал	**Опис**
Керування контентом	створення, зберігання та керування доступом до створення та використання навчальних матеріалів
Розробка та оформлення навчальних планів	планування розкладу занять, та персоналізація навчального досвіду
Залучення здобувачів вищої освіти та адміністрації	керування доступом до навчальних матеріалів, ресурсів та до моніторингу прогресу і результатів навчання
Спілкування та співпраця	електронна пошта, повідомлення, чати, блоги та ін.

Комунікація в реальному часі	відео або аудіо конференції

Крім елементів, що реалізовують зазначені вище основні функціональні можливості, до складу віртуальних освітніх середовищ включаються наступні компоненти (таблиця 1.2)

Функціональність віртуального освітнього середовища не розробляється під якийсь один конкретний навчальний курс, а для максимально повного охоплення усіх академічних програм, представлених у закладі освіти, надаючи доступ до них як співробітникам самого закладу, а також іншими організаціям, державним установам, які можуть використовувати систему [37-38].

Таблиця 1.2.

Перелік типових компонент віртуального освітнього середовища

Компонент	**Опис**
Силабуси навчальних курсів	Структурована навчальна програма відповідної дисципліни, що оновлюється на початок кожного навчального року
Зміст та навчальні матеріали курсів	Матеріали лекцій у вигляді відео, аудіо, текстових матеріалів, а також підтримку візуальних презентацій
Адміністративна інформація, що стосується курсу	Передумови (головним чином – це компетенції, необхідні для початку вивчення курсу), кількість кредитів, годин навчальних занять
Дошка повідомлень	Актуальна інформація о курсах, що проходять у поточний час
Додаткові матеріали	Навчальні матеріали розміщені у самій системі, або у вигляді покликань на сторонні ресурси

Матеріали для самоконтролю	Завдання здобувачам вищої освіти , тести
Нормативні форми контролю	Екзамени, заліки, диференційовані заліки, дипломи, виробнича практика, та ін.
Підтримка різних систем комунікації	Електронна пошта, корпоративні месенджери, спеціалізовані чати, або навіть сторінки у певних соціальних мережах
Адміністративний модуль	Керування розподілом ролей та прав доступу для науково-педагогічні працівників, асистентів, адміністрації факультету або університету
Документи	Система документообігу закладу освіти

Головні цілі використання віртуальних освітніх середовищ з точки зору закладів вищої освіти, та закладів післядипломної освіти представлені на рис.1.3.

Із боку **соціальної складової навчання:** навчання та взаємодія з однолітками підвищує рівень залученості та утримання здобувачів вищої освіти. Створення віртуального освітнього середовища забезпечує оптимізацію процесу комунікації між учасниками навчального процесу за

Рисунок 1.3. Цілі віртуальних освітніх середовищ для закладів вищої освіти

допомогою соціальних функцій, оскільки головними цілями середовища є відтворення аспектів аудиторного навчання та оптимізація його за допомогою інформаційно-комунікаційних технологій [35].

Види технологій взаємодії науково-педагогічні працівників та здобувачів вищої освіти представлені на рис.1.4.

Задачі соціальних функцій:

- створення відчуття спільності серед користувачів віртуального навчального середовища;
- забезпечення інструментів створення простору для спільного навчання.

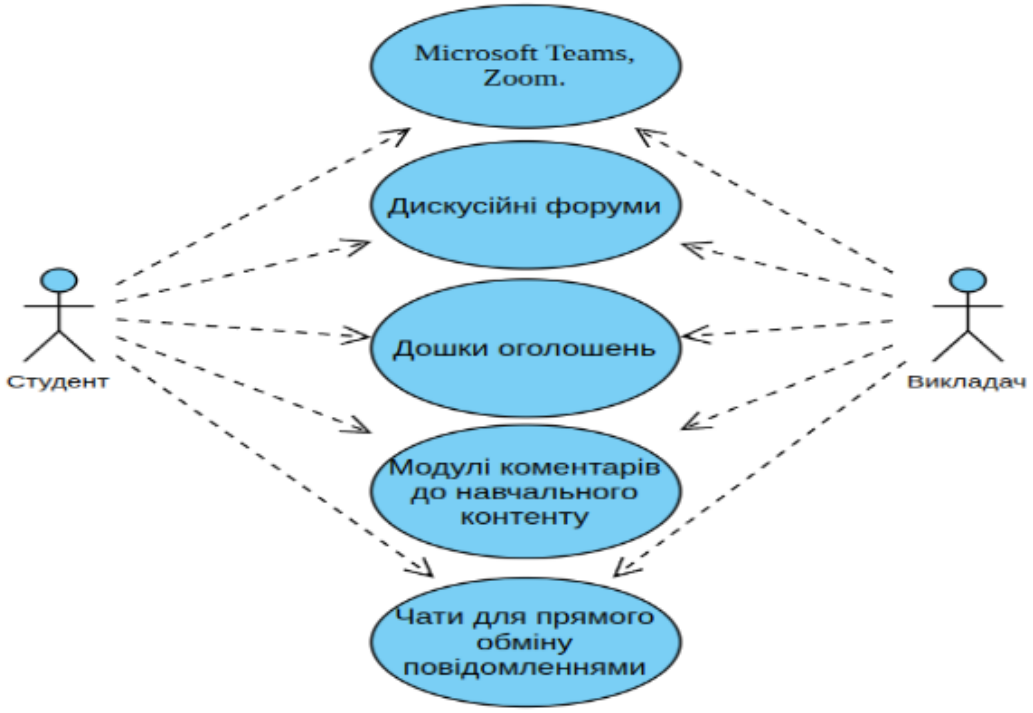


Рисунок 1.4. Технології підтримки соціальних функцій у віртуальному освітньому середовищі

Наприклад: заохочення написання відгуків користувачами. Відгуки користувачів про віртуальне освітнє середовище рис.1.4, надають адміністраторам та викладачам інформацію для оптимізації процесу навчання [36].

Методи вдосконалення навчального процесу вказані в таблиці 1.3.

Таблиця 1.3.

Методи реалізації ефективного навчального досвіду

Техніка мікронавчання	Заняття складається з коротких лаконічних частин, які зазвичай займають менше 10-15 хвилин. Наприклад, викладач може викласти курс за допомогою 10-хвилинного відео, після якого йде короткий розділ для читання [36].
Запровадження гейміфікації	Навчання складається із занять у стилі гри та онлайн-ігор, наприклад, зіставлення слова з визначенням або керованого моделювання.
Моніторинг прогресу здобувачів вищої освіти	Якщо студент завершує онлайн-заняття без зворотного зв'язку чи вказівок від науково-педагогічні працівників/тренерів, це може призвести до відчуття ізольованості, особливо якщо у студента виникли проблеми з вмістом курсу [37].
Використання інструментів аналітики та звітності	Адміністратори можуть бачити прогрес здобувача вищої освіти в режимі реального часу, дивлячись на такі критерії, як показники завершення, прохідні бали, частота входу в систему, час, витрачений на індивідуальні заняття тощо.

Класичне автономне оцінювання під час завершення заняття не має інструментів, які забезпечують досягнення необхідного рівня інформативності у порівнянні із інструментами звітності віртуального освітнього середовища [40].

Для побудови ефективного віртуального освітнього середовища, спрямованого на задоволення максимально різноманітних вимог сучасного здобувача вищої освіти, необхідна надійна,

багатофункціональна та інтуїтивно зрозуміла програмна система або сукупність систем, що працюють разом, яка об’єднує усі види програмних інструментів та соціальних компонент, що утворюють віртуальне освітнє середовище (таблиці 1.2.), технології підтримки соціальних функцій (рис 1.3) та інструменти управління продуктивністю та керування контентом.

Найбільш розповсюдженим варіантом реалізації віртуального освітнього середовища є онлайн-портал [44], доступ до якого здійснюється через відповідні логін і пароль.

Коли здобувачі вищої освіти вперше потрапляють до закладу вищої освіти, вони отримують інформацію та інструкції щодо доступу до віртуального освітнього середовища.

Модель віртуального освітнього середовища конкретного закладу вищої освіти, будується із урахуванням унікальних функціональних вимог закладу [41]. Користувачами платформи є всі учасники навчального процесу: здобувачі вищої освіти, науково-педагогічні працівники, директори та керівництво, і адміністрація.

Функціональні вимоги віртуального освітнього середовища для здобувачів вищої освіти:

- забезпечення проведення віртуальних занять (в прямому ефірі чи в записі);
- надання доступу до навчальних матеріалів;
- надання доступу до додаткових матеріалів;
- надання доступу до контактних каналів;
- надання доступу до сторінок для подання вправ і проходження тестів;
- надання доступу до дискусійних форумів, чатів і коментарів щодо виконаної діяльності.

Функціональні вимоги віртуального освітнього середовища [35] для науково-педагогічні працівників:

- обмін основними і допоміжними матеріалами;

- керування даними та діяльністю здобувача вищої освіти;
- створення та застосовування оцінювання;
- додавання коментарів;
- проведення моніторингу розвитку кожного окремого здобувач вищої освіти.

Функціональні вимоги віртуального освітнього середовища для адміністраторів та керівництва закладу освіти:

- доступ інструментів контролю оцінювання;
- оцінювання критичних точок;
- пошук рішень для постійного вдосконалення процесу викладання.

Основні переваги віртуального освітнього середовища для закладу вищої освіти та його здобувач вищої освіти наведені в таблиці 1.4.

Таблиця 1.4.

Переваги використання віртуального освітнього середовища

Ставлення до здобувача вищої освіти як до «головного героя»	Здобувач вищої освіти має більше контролю над процесом навчання з використанням віртуального освітнього середовища. Це спонукає здобувач вищої освіти проявляти відповідальність і дисципліну, щоб планувати навчальний час і брати на себе лідерство у власної освіті [42].
Надання здобувачам вищої освіти інструментів для виправлення помилки	Здобувач вищої освіти може повторно переглядати матеріали лекцій та практичних або лабораторних занять. Вправи також можна спробувати виконати повторно. Здобувачі вищої освіти можуть отримати доступ до занять, коли вони більш зосереджені та вибрати найкращий час для навчання [44].

Доступ до звітів про успішність	Науково-педагогічні працівники мають доступ до вичерпних звітів про успішність, які містять інформацію про рівень залученості здобувач вищої освіти, найефективніші курси та більшість матеріалів, які можливо зберегти на власний гаджет [46]. Цей аналіз дає змогу зрозуміти сильні сторони закладу освіти, факультету, кафедри та здобувачів вищої освіти, виявити проблемні місця та розробити відповідні стратегії для вдосконалення навчального процесу.
Забезпечення індивідуального навчання	Звіти використовуються для налаштування курсів та покращення процесу навчання здобувачів вищої освіти [44]. Здобувачі вищої освіти самі можуть налаштовувати свій розклад, вибирати, які предмети вивчати першими та який контент споживати.
Адаптація матеріалів під різні екрани	Віртуальні освітні середовища розробляються адаптивними, що підлаштовуються під різні розміри екрана. Це дозволяє здобувачам вищої освіти використовувати для навчання комп’ютери, планшети чи мобільні телефони.
Забезпечення безпеки даних	Різні рівні доступу до віртуального освітнього середовища забезпечують контроль над тим, хто використовує платформу [45]. Для захисту інформації здобувачів вищої освіти, науково-педагогічні працівників і самого закладу освіти можна застосувати кілька вимог безпеки.

Підтримка очних занять	Підтримка віртуальними освітніми середовищами очних занять. Навіть у курсах, що використовують звичайні моделі, віртуальні навчальні середовища можуть сприяти застосуванню онлайн-контенту та педагогічним інноваціям, покращуючи навчальний процес

У теорії сучасної віртуальної освіти виділяються три типи віртуальних освітніх середовищ (таблиця1.5.), основна відмінність між якими полягає в рівні взаємодії між викладачами та здобувачами вищої освіти.

Таблиця 1.5.

Типи віртуальних освітніх середовищ

Синхронне віртуальне освітнє середовище	Синхронне віртуальне освітнє середовище зосереджується на цифрових заняттях, що транслюються в прямому ефірі, щоб представити навчальний матеріал. Зазвичай це забезпечує вищий рівень взаємодії між здобувачами вищої освіти та викладачами, оскільки вони можуть взаємодіяти практично в реальному часі [48]. Здобувачі вищої освіти отримують миттєвий зворотній зв’язок від своїх науково-педагогічних працівників, а науково-педагогічні працівники можуть оцінити здібності своїх здобувачів вищої освіти під час проходження курсу. Науково-педагогічні працівники та здобувачі вищої освіти входять у віртуальні чати для своїх груп. Це дуже схоже на особисту аудиторію, де здобувачі вищої освіти покладаються на інструктора, щоб вивчити зміст курсу. Синхронне навчання ідеально

	підходить для людей, які віддають перевагу викладанню в реальному часі та постійному незмінному темпі навчання. Це також забезпечує більш практичний досвід для невеликих груп, оскільки здобувачі вищої освіти можуть взаємодіяти один з одним для співпраці.
Середовища для асинхронного віртуального навчання.	Асинхронне віртуальне освітнє середовище включає попередньо записані курси, до яких здобувачі вищої освіти можуть отримати доступ у власному темпі. Науково-педагогічні працівники та здобувачі вищої освіти підключаються до платформ віртуального освітнього середовища через системи обміну повідомленнями або дискусійні форуми, але зміст курсу подається у вигляді готових лекцій [48]. Хоча цей тип віртуального освітнього середовища обмежує співпрацю, він ідеальний для здобувачів вищої освіти, які віддають перевагу навчанню у вільний час . Більшість платформ віртуального освітнього середовища, призначених для вищої освіти, пропонують асинхронне навчання для здобувачів вищої освіти, особливо для здобувачів, які можуть не мати таких самих можливостей для відвідування занять, як їхні науково-педагогічні працівники. Асинхронні освітні середовища дозволяють здобувачам вищої освіти і викладачам проводити заняття, незважаючи на різницю в часі та графіки доступності. Наприкінці курсу здобувачі вищої освіти часто проходять опитування, щоб оцінити свій прогрес у навчанні [49]. Платформи віртуального освітнього середовища зазвичай мають вбудовані інструменти оцінювання для науково-педагогічних працівників, які пропонують тести

	для перевірки знань своїх здобувачів вищої освіти. Вони також можуть спілкуватися зі своїми здобувачами вищої освіти за допомогою програмних засобів для онлайн-спілкування, щоб вирішувати питання, запити чи занепокоєння.
Середовища для гібридного віртуального навчання.	Налаштування гібридного навчання полягає у поєднанні занять у прямому ефірі з попередньо записаними лекціями. Цей тип віртуального освітнього середовища ідеально підходить для тих, хто хоче налаштувати свій навчальний процес, зберігаючи при цьому структуроване освітнє середовище, яке забезпечує заняття в режимі реального часу [49]. Науково-педагогічні працівники публікують відеолекції, презентації та інші навчальні матеріали для здобувачів вищої освіти до або після віртуальних занять, проводять оцінювання за допомогою синхронних або асинхронних налаштувань залежно від стилю викладання викладача чи уподобань здобувачів вищої освіти щодо навчання. Гібридне освітнє середовище пропонує студентам можливості, як для налаштування, так і для практичного навчання. Багато платформ дистанційного навчання пропонують гібридні віртуальні навчальні середовища як керовані освітні середовища для здобувачів вищої освіти і науково-педагогічних працівників, щоб розробити різні форми проведення занять.

Віртуальне освітнє середовище має сукупність переваг та недоліків рис. 1.5 в залежності від уподобань і ресурсів, доступних для здобувача вищої освіти.

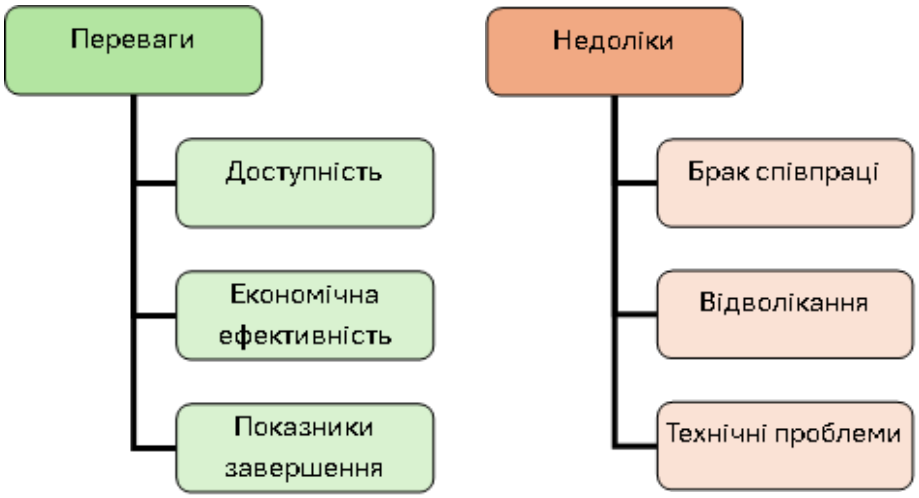


Рисунок 1.5. Переваги та недоліки використання віртуального освітнього середовища

Основні переваги та недоліки віртуальних освітніх середовищ [43] наведено у таблицях 1.6 та 1.7.

Таблиця 1.6.

Переваги віртуальних освітніх середовищ

Доступність	Основним чинником віртуальних освітніх середовищ є доступність. Здобувачі вищої освіти, які не мають ресурсів або часу для відвідування очних занять, можуть отримати належну освіту за допомогою віртуального навчання. Віртуальне освітнє середовище ідеально підходить для охоплення здобувачів вищої освіти, які можуть мати проблеми з мобільністю, працюють віддалено або мають інші обмеження, через які їм важко відвідувати заняття фізично [51]. Так само науково-педагогічні працівники можуть легко отримати доступ до процесу навчання своїх здобувачів вищої освіти за допомогою зручної системи управління навчанням, яка консолідує дані здобувачів вищої освіти. Науково-педагогічні працівники можуть відповідним чином

	коригувати свої методи навчання, не стикаючись із труднощами очних занять.
Економічна ефективність	Курси електронного навчання часто є економічно ефективними, особливо для установ, які прагнуть навчати великі групи здобувачів вищої освіти. За допомогою лише однієї платформи віртуального освітнього середовища науково-педагогічні працівники можуть охопити кількох здобувачів вищої освіти у різних областях. Платформи також мають групові плани для великих команд, що робить віртуальне освітнє середовище більш економічно ефективним рішенням для навчання порівняно з фізичними групами, для яких потрібні приміщення.
Закінченість	Онлайн-навчання призводить до високих показників завершення процесу серед здобувачів вищої освіти. Оскільки студенти можуть вивчати матеріал курсу у вільний час, вони, швидше за все, успішно складуть курс [52]. Цей персоналізований підхід до управління контентом дозволяє студентам і викладачам оптимізувати свої навчальні стратегії.

Таблиця 1.7.

Недоліки віртуальних освітніх середовищ

Брак співпраці	Віртуальні світи відрізняються від реального життя, те саме стосується віртуального освітнього середовища та фізичного навчання. У віртуальному освітньому середовищі здобувачі вищої освіти мають обмежені можливості для співпраці та спілкування з одногрупниками. Деяким здобувачам вищої освіти

	може навіть бути важко спілкуватися з іншими здобувачами вищої освіти для обговорення занять і тестів [53]. Брак співпраці може перешкоджати соціальному розвитку людини та навичкам співпраці.
Відволікання	Віртуальне освітнє середовище дає змогу проводити заняття з будь-якого місця, що є великою перевагою. Проте здобувачам вищої освіти може бути важко зосередитися на матеріалі курсу, коли в їхньому оточенні присутні фактори, які відволікають увагу. На відміну від навчання в аудиторії, під час якого відволікання можна контролювати та усунути, навчання з різних місць може заважати здобувачам вищої освіти опановувати знаннями, не даючи їм зануритися в предмет.
Технічні проблеми	Оскільки віртуальне освітнє середовище значною мірою покладається на цифрові аспекти, будь-які технологічні проблеми можуть спричинити перебої в процесі навчання [54]. Нестабільні підключення до інтернету, виникнення програмних збоїв платформи та проблеми з пристроєм можуть негативно вплинути на прогрес здобувача вищої освіти.

Під час проходження здобувачами вищої освіти виробничої практики виділяються три напрями до використання віртуальних освітніх середовищ [46-47]:

1. **Професійно-технічна освіта та підготовка.** Цей напрямок спрямований на підготовку здобувачів вищої освіти до роботи у відповідних професійно-технічних установах з метою отримання необхідних умінь, навичок і досвіду. Велика кількість професійно-

технічних закладів перейшли до дистанційного навчання із використанням віртуальних освітніх середовищ для проведення курсів для здобувачів вищої освіти у будь-який час і в будь-якому місці.

2. **Передовий тренінг.** За цим напрямком практиканти розподілені по різних місцях, де працює організація. Через це, вони повинні мати знання про стандартні протоколи охорони здоров'я та безпеки на роботі [46]. Віртуальне освітнє середовище надає організаціям інструменти для організації та оптимізації навчального процесу «без відриву від роботи» для працівників.

3. **Навчання при віддаленій роботі**. Напрямок, призначений для працівників, які не мають визначеного робочого місця. Віртуальне освітнє середовище забезпечує для організації підтримку узгодженості у навчанні на робочому місці для працівників, що працюють віддалено, через надання інструментів віддаленого розгортання відповідного навчального контенту. Тренінг-менеджери отримують інструменти та засоби для перегляду прогресу своїх співробітників, та проведення стандартизованого та неупередженого оцінювання [47]. В IT-індустрії для керування інфраструктурою організацій, налагодження взаємодії як між структурними підрозділами самої організації, так і з її клієнтами використовують *системи управління взаємовідносинами* (англ. Customer relationship management).

1.4. Система управління відносинами з клієнтами у сучасному віртуальному освітньому середовищі

Функціональні можливості систем управління відносинами зосереджуються на збиранні та аналізу інформації з великої кількості різноманітних джерел рис. 1.6 для надання компаніям максимально великої кількості корисної інформації про відповідну цільову аудиторію, та про найефективніші засоби задоволення її потреб [50-53].

Системи управління відносинами використовуються для роботи з

минулими, поточними або потенційними клієнтами. Структурно вони є сукупністю концепцій, процедур і правил, яких дотримується установа під час взаємодії зі своїми користувачами та програмні системи, які забезпечують повний цикл взаємодії. Цей повний цикл охоплює наступні варіанти взаємодії:

- прямі контакти з клієнтами;
- продажі та операції, пов'язані з обслуговуванням;
- прогнозування та аналіз моделей і поведінки споживачів з точки зору компанії.

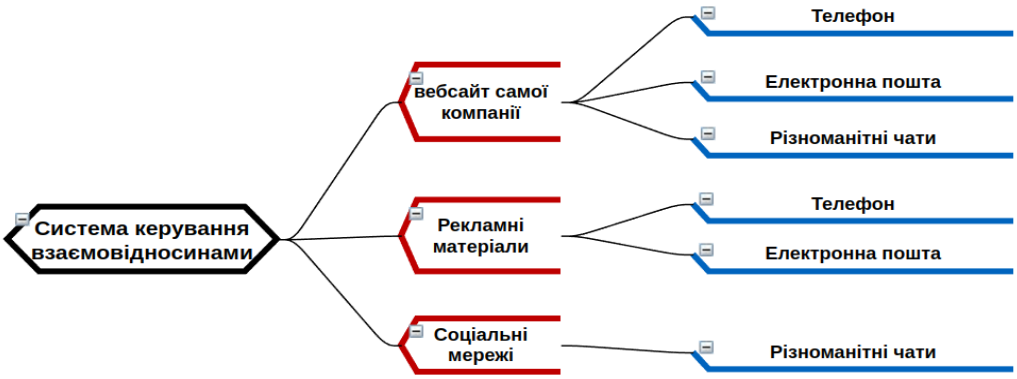


Рисунок 1.6. Джерела інформації для систем керування взаємовідносинами

Системи управління відносинами поділяються на наступні категорії, що представлено на рисунку 1.7:

Стратегічні системи – це системи, при побудові яких головна увага приділяється інфраструктурі взаємодії з клієнтами.

Операційні системи – це системи, головною метою яких є інтеграція та автоматизація бізнес-процесів клієнтів. Складовими частинами цих систем є наступні компоненти [92]:

- інформаційна панель, задача якої — надання узагальненої інформації про основні функціональні можливості клієнта й історію його взаємодії з системою та іншими клієнтами;
- окрема сторінка для кожного клієнта компанії.

Операційна система управління відносинами складається з трьох основних компонент:

- система автоматизації продажів;
- система автоматизації маркетингу;
- системи автоматизації обслуговування.

Аналітичні системи аналізують інформацію про клієнтів, що надходить із різноманітних джерел, і представляють її у відповідному вигляді, необхідному бізнес-менеджерам для обґрунтування прийняття рішень. До головних методів аналізу аналітичних систем управління відносинами належать:

- аналіз даних;
- кореляція;
- розпізнавання шаблонів.

Результати аналізу спрямовуються та оптимізацію процесів обслуговування клієнтів наступними шляхами:

- шляхом знаходження проблеми клієнта, що вирішуються засобами аналітичної системи;
- шляхом диференціації маркетингових підходів для різних груп та категорій споживачів.

Платформи клієнтських даних (англ. Customer data platform, CDP) – це сукупність засобів програмного забезпечення, спрямованих на створення постійної, єдиної бази даних про клієнтів. Інформація отримується із переліку доступних платформі джерел, аналізується та об’єднується для створення єдиного профілю клієнта.

Головні функціональні класи систем управління відносинами наведено на рисунку 1.7. Побудова та управління взаємодією з клієнтами за допомогою інструментів, притаманних маркетингу, моніторингу відносин, коли і як вони розвиваються протягом певних окремих етапів, управління цими змінами на усіх етапах та прийняття того, що розподілення цінності відносин для установи не є однорідним [95]

шляхом розбудови та керування відносинами з клієнтами засобами та функціями інструментів маркетингу, установи отримують перевагу над конкурентами за рахунок наступних процесів, спрямованих на оптимізацію та автоматизацію охоплення потреб клієнтів:

організації архітектурного дизайну;

- застосування схем стимулювання;
- структуризації груп клієнтів;
- визначенню окремих етапів, у вигляді пов'язаних транзакцій;
- ведення обліку успішності відносин із конкретним клієнтом.

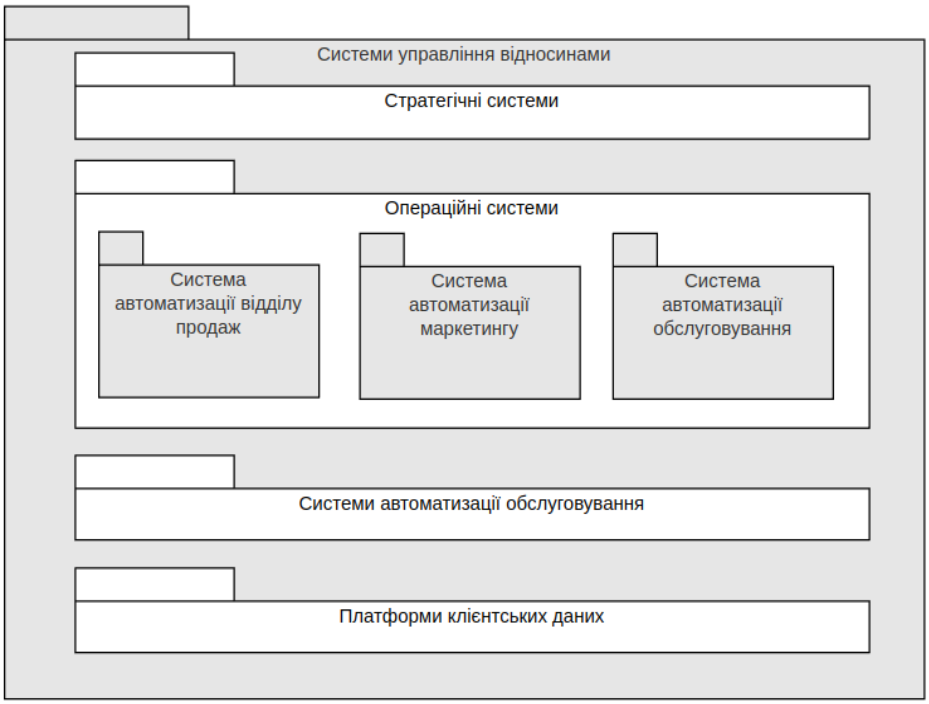


Рисунок 1.7. Класифікація систем управління відносинами

З точки зору освітнього середовища, використання систем керування відносинами в закладах середньої та вищої освіти надає безліч переваг рисунок 1.8. Побудована на основі вимог до системи керування відносинами навчальна система забезпечує уніфікований і прозорий доступ для здобувачів вищої освіти, також надаючи наступні переваги співробітникам закладів вищої освіти [94]:

- організація контактних даних здобувачів вищої освіти;
- аналіз процесу навчання;

- доступ до операційних і адміністративних даних.

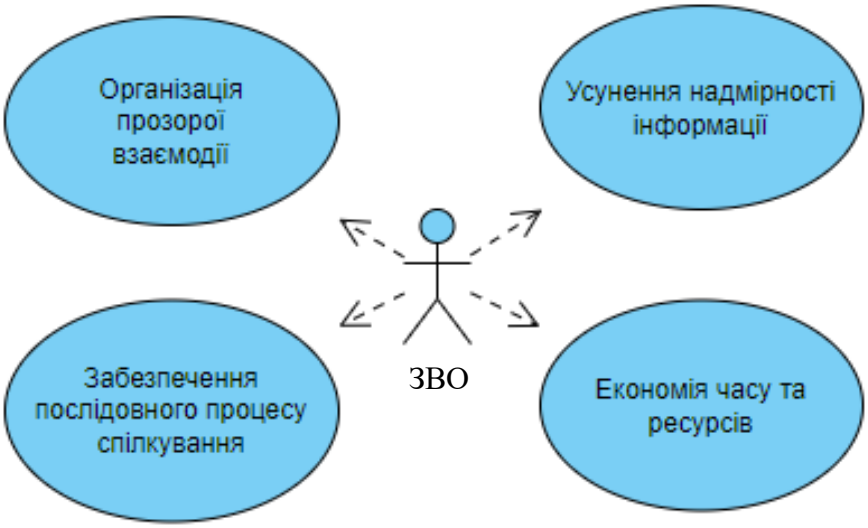


Рисунок 1.8. Переваги використання систем управління відносинами з точки зору закладів вищої освіти

Багато закладів вищої освіти використовують автоматизовані системи керування відносинами з певних причин рис 1.9.

Заклади освіти будь-якого рівня: школи, коледжі чи університети, потребують механізми та засоби для упорядкування та обробки всього цього великого обсягу інформації в єдиному місці для доступу учасників освітнього процесу до тих даних, які їм необхідні, та для проведення комплексного аналізу.

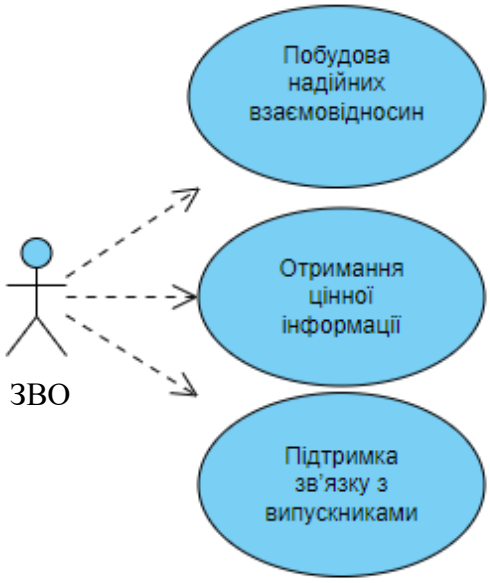


Рисунок 1.9. Причини використання систем управління відносинами з точки зору закладів вищої освіти

Головною перевагою впровадження систем управління взаємовідносинами для закладів вищої освіти — оптимізація процесу залучення нових здобувачів вищої освіти та одночасне зміцнення відносин із поточними здобувачами вищої освіти, та випускниками. Основні переваги освітньої системи управління взаємовідносинами вказані на рис 1.10.Системи управління взаємовідносинами з клієнтами надають закладам освіти різного рівня інструменти для керування відносинами усіма категоріями користувачів системи [95]:

- здобувачі вищої освіти;
- батьки;
- науково-педагогічний персонал;
- випускники;
- компанії, роботодавці, як державного, так і приватного секторів.

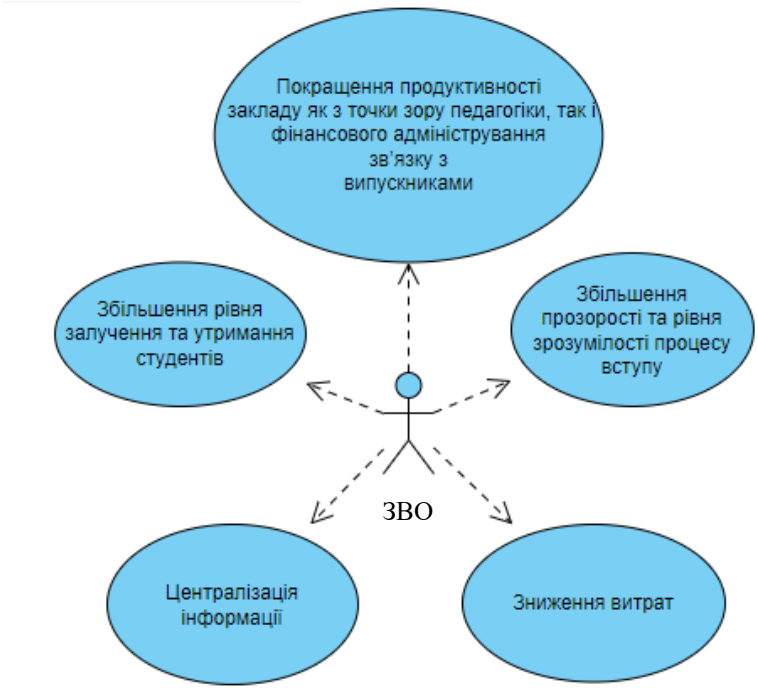


Рисунок 1.10. Переваги освітньої системи керування взаємовідносинами

Системи управління взаємовідносинами з клієнтами надають інструменти для адміністрування аспектів всього життєвого циклу

здобувача вищої освіти від зарахування та відстеження академічного прогресу до управління фінансовою частиною навчального процесу та інструментами генерації звітів. У будь-якому випадку головна мета полягає в покращені комунікації, ефективності, і, зрештою, результативністю процесу навчання шляхом забезпечення усіх учасників навчального процесу необхідною інформацією [96].

Використання системи управління взаємовідносинами надає суттєві *переваги* у порівнянні із традиційними методами організації навчального процесу рис.1.11.

Складові частини систем управління взаємовідносинами для кожного окремо взятого закладу вищої освіти можуть суттєво відрізнятись, беручі до уваги перелік спеціальностей, які викладаються на тій чи іншій кафедрі, особливості організації студентської практики тощо.

Рисунок 1.11. Переваги систем керування взаємовідносинами над традиційними підходами до організації навчального процесу

Програмне забезпечення освітньої системи керування взаємовідносинами відіграє важливу роль у процесах розвитку і функціонування закладу освіти:

Основні функціональні складові частини типової системи керування взаємовідносинами зазначено на рис 1.12.

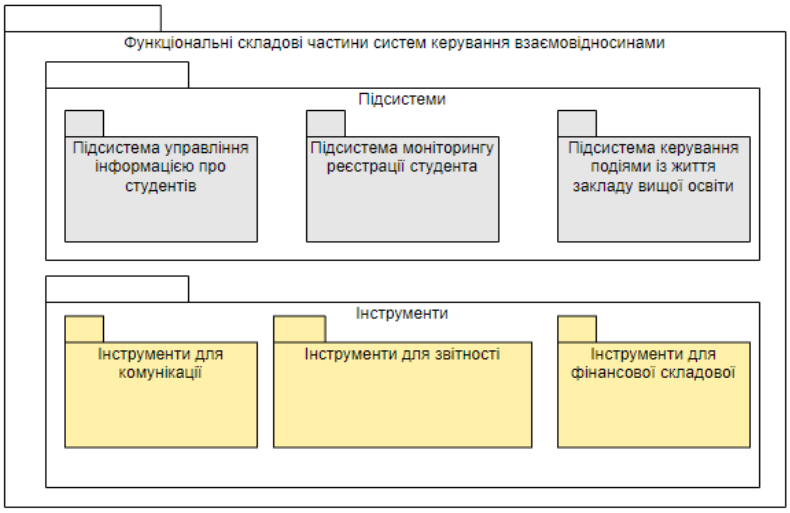


Рисунок 1.12. Функціональні підсистеми керування взаємовідносинами

1. **Забезпечення ефективного процесу ведення супровідної документації.** Програмні складові системи керування взаємовідносинами спрямовані на [95]:

- збір важливої інформації від майбутніх здобувачів вищої освіти;
- збереження цієї інформації в централізованій базі даних, до якої персонал приймальної комісії має доступ;
- оптимізація процесів виявлення кращих студентів викладачами та адміністрації з підбору персоналу;
- об’єднання повторюваних профілів студентів та призначення кожному студенту унікального номера вступу.

2. **Оптимізація процесів обробки запитів.** До складу систем керування взаємовідносинами входять необхідні інструменти для взаємодії та підтримки здобувачів вищої освіти, у яких виникають труднощі на певних етапах навчання. Наприклад: готові відповіді – короткі блоки тексту, які можна багаторазово використовувати, для оптимізації обміну інформацією між адміністрацією закладу вищої освіти та студентами. Якщо серед готових відповідей немає тієї, яка відповідає

запиту здобувача вищої освіти, то до нього призначають відповідно співробітника адміністрації або інших спеціалістів [96].

3. **Збільшення кількості вступників.** Модулі підтримки процесів організації ефективних акцій набору здобувачів вищої освіти надають науково-педагогічним працівникам, іншим співробітникам та успішним випускникам інструменти для підключення своїх сторінок в соціальних мережах та інших маркетингових матеріалів до системи керування взаємовідносинами, для координації всієї маркетингової діяльності кафедр та факультетів через інтерфейси системи керування взаємовідносинами. Визначаючи пріоритетні канали, які дають найкращу віддачу від інвестицій, заклад вищої освіти збільшує кількість здобувачів, які навчаються очно [97].

4. **Інтеграція із зовнішніми програмними середовищами.** Найкращі освітні системи керування взаємовідносинами підтримують безліч інтеграцій. Це дозволяє відділу з підбору персоналу розширити функції системи керування взаємовідносинами за межі її основної функціональності. Наприклад, можливо інтегрувати власну систему керування взаємовідносинами із соціальними мережами, що дозволить керувати та відстежувати свої маркетингові зусилля там. Також можливо інтегрувати власну систему керування взаємовідносинами з обліковими записами електронної пошти. Таким чином, можливо миттєво отримувати системні електронні листи та швидко відповідати на них [98].

5. **Підтримка співпраці та командної роботи.** Ефективна співпраця між командами та відділами є обов’язковою для роботи закладу вищої освіти. Використання розподіленого сховища інформації у програмному забезпеченні системи керування взаємовідносинами є основною системною вимогою забезпечення безперебійної командної роботи:

- члени команди мають функції обміну даними між відділами різного рівня;

- відділи співпрацюють один з одним для отримання успішних результатів у пересічних завданнях.

Таким чином, розподілене сховище оптимізує процеси у навчальній установі.

6**. Організація аналітики для освітньої установи.** Система керування взаємовідносинами надає можливість створювати настроюванні звіти, які вимірюватимуть рейтинг навчальної установи в будь-який час. Співробітники можуть отримувати корисну інформацію з величезних баз даних і використовувати ці показники для оцінки моделей успіху та прийняття рішень, які підвищать утримання студентів.

Підводячи підсумки зазначаємо, що, з точки зору професорсько-викладацького складу є можливість перейти до програмного середовища орієнтованого на здобувачів вищої освіти.

1.5. Студентська інформаційна система

Студентська інформаційна система (Student Information System) – комплекс програмного забезпечення, який надає закладам вищої освіти функціональні інструменти для процесів [102]:

- оцифрування інформації про здобувачів вищої освіти;
- ефективного керування отриманою інформацією.

Система надає закладам вищої освіти усіх рівнів функціональні можливості обробки інформації про здобувачів вищої освіти через мережу «Інтернет» для:

- автоматизації адміністративних процесів;
- автоматизації академічних процесів.

Завдяки студентській інформаційній, системі школи, ліцеї, гімназії та заклади вищої освіти будь-якого рівня можуть автоматизувати великий спектр адміністративних, навчальних функцій та забезпечити їх набагато ефективніше, ніж без неї. Головні функціональні можливості студентської інформаційної системи:

- реєстрація здобувачів вищої освіти на заняття;
- формування розкладів;
- відстеження відвідуваності;
- зберігання звітів про успішність, таких як оцінки та досягнення (участь конференціях, проходження зовнішніх онлайн-курсів, участь в олімпіадах тощо)

Студентська інформаційна система надає закладам вищої освіти всі необхідні функції для автоматизації навчальних процесів і конкретних задач. Автоматизація спрямована на підтримку та стимулювання навчання здобувачів вищої освіти з першого дня й до моменту випуску.

Функціональність студентської інформаційної системи спрямована на підтримку та оптимізацію процесів керування різними аспектами життєвого циклу закладу вищої освіти [103].

Більшість функцій, включених в платформу студентської інформаційної системи, реалізуються у вигляді окремих програмних застосунків.

Ефективно організована та багатофункціональна студентська інформаційна система підтримує та сприяє автономії закладу вищої освіти. Підсистеми типової студентської інформаційної системи реалізують наступні *функції (рис 1.13.)*, спрямовані на оптимізацію освітніх процесів.

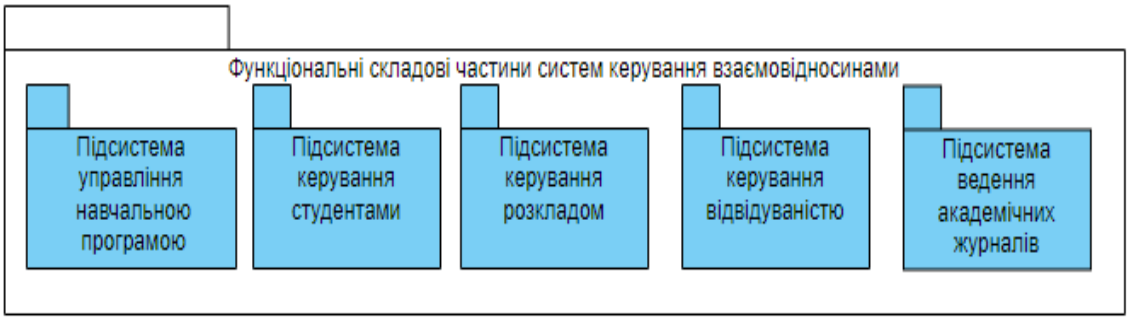


Рисунок 1.13. Підсистеми типової студентської інформаційної системи

Управління навчальною програмою (або управління навчальним планом) - процес, необхідний для формування траєкторії навчання для студентів в освітній установі. Процес управління навчальною програмою школи визначає навчальні цілі та освітні очікування для здобувачів вищої освіти, які в більшості випадків представлені у вигляді курсів, структурованих за категоріями різного рівня [104]. Процес організації структури навчальної програми складається з ряду складних кроків:

- прийняття рішень щодо того, які основні дисципліни мають бути додані до навчальної програми;
- послідовне представлення навчального плану для забезпечення його подальшої модифікації.

Студентська інформаційна система надає директорам та членам адміністрації закладу вищої освіти, які відповідають за процеси управління закладом, засоби представлення ієрархії категорій інформації про здобувачів вищої освіти лише кількома простими діями. Функціональні можливості підсистеми керування навчальними програмами спрямовані побудову необхідної структури для організації та виконання адміністративних та освітніх завдань [104-105].

Керування здобувачами вищої освіти — основна функціональна можливість адміністрації закладу освіти. Програмне середовище, яке надає адміністраторам наступні функції для зберігання, отримування та обробки інформації про студентів, є надзвичайно важливою для всіх закладів вищої освіти; незалежно від їх типу чи розміру. Ефективне управління інформацією здобувачів вищої освіти передбачає організацію процесу зберігання повного спектру необхідної інформації:

- *студентська інформаційна система* надає адміністраторам закладу вищої освіти функцій для зберігання повної інформації про здобувачів вищої освіти у вигляді онлайн-профілів.

У свою чергу, *студентська інформаційна система* надає викладачам та адміністрації інструменти для оптимізації процесів виконання основних навчальних завдань:

- засоби реєстрації здобувачів вищої освіти на поточний навчальний період;
- засоби студентів на відповідні навчальні дисципліни;
- засоби для стягнення із здобувачів вищої освіти плати за навчання;
- адміністратори мають інструменти для імпорту записів;
- засоби перегляду інформації про здобувачів вищої освіти із застосуванням різноманітних фільтрів;
- засоби створення та керування академічним розкладом.
- *Управління академічним розкладом* відноситься до найбільш складних задач для адміністраторів закладів освіти будь-якого рівня. Функціональність студентських інформаційних систем, надає закладам вищої освіти інструменти для створення та розміщення академічного розкладу онлайн.

Щодо *керування відвідуваністю,* то це безперервний процес точного та ефективного ведення журналів відстеження відвідування здобувачами вищої освіти навчальних занять, які використовуються для розрахунку або коригування плати за навчання, науково-педагогічні працівники та/або секретарі можуть вчасно відстежувати відвідуваність; і своєчасно зберігати цю інформацію в мережі «Інтернеті».

Ведення академічних журналів. Здебільшого в освітніх установах досі використовуються паперові академічні журнали. Це має наступні *недоліки:*

- викладачам потрібно витрачати додатковий час, для створення зазначених журналів з оцінками і правильно їх заповнювати;
- викладачам необхідно потрібно носити з собою друковану копію кожного, якщо вони знадобляться.

Завдяки студентській інформаційній системі процес керування академічним журналом стає набагато ефективнішим. Адміністраторам потрібно лише кілька хвилин, щоб створити необхідну кількість журналів оцінок для всіх академічних груп, додати дату та коментарі – при необхідності.

Студентська інформаційна система надає наступні *переваги [106]*:

- забезпечення ефективного, автоматизованого процесу ведення академічних журналів для науково-педагогічних працівників і для адміністрації;
- засоби організації доступу до журналу для кожної зацікавленої сторони: адміністрації, науково-педагогічним працівникам, ректорату, здобувачам вищої освіти та батькам.

Процес отримання актуальної інформації відносно поточного навчального процесу, в студентській інформаційній системі організовано через відповідні інструменти, що вимагає від здобувача вищої освіти наявності персонального облікового запису для входу в студентську інформаційну систему закладу вищої освіти [107]. Обліковий запис студента надає наступні функціональні можливості:

- доступ до перегляду власного онлайн-профілю;
- доступ до інструментів відстеження змін навчального розкладу та оголошень;
- перегляд власних оцінок успішності;
- доступ до інструментів зв’язку з викладачами, одногрупниками та адміністрацією.

Студентська інформаційна система, яка містить портал для науково-педагогічних працівників, надає викладачам наступні функціональні можливості:

- виконання адміністративних задач;
- ведення журналів;
- перегляд профілів здобувачів вищої освіти;

- відстеження відвідуваності;
- доступ до інструментів зв’язку з адміністрацією;
- доступ до інструментів зв’язку з іншими викладачами.

Ґрунтуючись на досвіді побудови студентських інформаційних систем переходимо до дослідження систем керування процесом навчання.

1.6. Системи керування навчанням у сучасному віртуальному освітньому середовищі

Для керування та підтримки безпосередньо самих освітніх курсів у віртуальних освітніх середовищах використовуються системи керування навчанням (LMS) [78] – програмні середовища призначені для:

- адміністрування навчальних курсів та матеріалів;
- документування навчальних курсів та матеріалів;
- відстеження рівня проходження здобувачами вищої освіти навчальних курсів та матеріалів;
- створення звітів по відповідним навчальним курсам та матеріалам;
- надання навчальних курсів та матеріалів.

Система керування процесом навчанням забезпечує доступ до усіх типів навчальних матеріалів, що включають відео, курси, семінари та документи. Системи керування навчанням складають найбільший сегмент ринку систем навчання. Вперше системи керування навчанням почали активно використовуватись наприкінці 1990-х років [79]. Проте масове зростання використання систем керування навчанням почалось через наголос на дистанційному навчанні під час пандемії COVID-19.

Головне призначення розробки систем керування навчанням полягає у виявленні прогалин у навчальному процесі, застосовуючи аналітичні засоби та інструменти для звітності [80]. Системи керування навчанням зосереджуються на підтримці дистанційного освітнього процесу, але підтримують низку застосувань, діючи як платформа для

розміщення навчальних матеріалів онлайн, включаючи курси, як асинхронні, так і синхронні. Основною моделлю застосування систем управління навчанням у сфері вищої освіти — є *перевернутий клас.* Сучасні системи управління навчанням включають інтелектуальні алгоритми для створення автоматичних рекомендацій для курсів що базуються на профілі та навичках користувача, а також на інформації із навчальних матеріалів, для того щоб рекомендації були ще точнішими [81].

Технічно система управління навчанням може бути розміщена локально або у постачальника. Хмарна система, розміщена постачальником, має архітектуру System as Service (англ. програмне забезпечення як послуга). Збереження усіх даних в системі, розміщеній у постачальника, забезпечується постачальником, а користувачі отримують доступ до них через інтернет, на персональному комп'ютері або мобільному пристрої. Системи, розміщені у постачальників, потребують менше технічних навичок з боку користувачів системи [100].

Рисунок 1.14. Функціональні можливості систем управління навчанням з точки зору викладача

Заклади вищої освіти користуються системами управління навчанням, розміщеними локально, оскільки це дозволяє організувати процеси розміщення та керування усіх даних системи на внутрішніх

серверах, що надає співробітникам закладу функціональні можливості для модифікації та підтримки програмного забезпечення внутрішньою командою [83]. Окремі особи та організації, як правило, дотримуються хмарних систем через вартість внутрішнього хостингу та обслуговування.

Основні переваги систем управління навчанням [84]:

- надання навчального контенту та інструментів безпосередньо здобувачам вищої освіти;
- автоматизація процесів оцінювання;
- створення та підтримка спеціалізованих груп користувачів;
- створення прозорої моделі комунікації між здобувачами вищої освіти та науково-педагогічними працівниками;
- інструменти для оптимізації процесу онлайн-навчання;
- слідкування за прогресом навчання;

Подібні системи надають вбудовані функції з великою кількістю налаштувань [99]:

- *здобувачі вищої освіти* отримують інструменти для відстеження свого прогресу в режимі реального часу;
- *науково-педагогічні працівники* отримують інструменти для контролю та повідомлення про ефективність процесу навчання (рис 1.14).

Таблиця 1.8

Основні функціональні можливості систем керування навчанням

Функція	**Опис та призначення**
Керування курсами, користувачами та ролями	Системи управління навчанням використовуються для створення структурованого змісту курсу. Науково-педагогічний працівник може додавати текст, зображення, відео, PDF-файли, покликання на інтернет ресурси, табличну інформацію, різноманітні тести, слайд-шоу тощо. Також, науково-педагогічний працівник має права для створення різних типів

	користувачів: редакторів, відвідувачів, інших науково-педагогічних працівників, здобувачів вищої освіти та батьків. Це допомагає контролювати, до якого матеріалу здобувачі вищої освіти мають доступ, відстежувати прогрес у навчанні та залучати здобувачів вищої освіти за допомогою інструментів для зв'язку. Система надає науково-педагогічним працівникам можливості та інструменти для керування курсами та модулями, для зарахування здобувачів вищої освіти або для надання їм можливість реєструватись на курс самостійно [87].
Онлайн оцінювання	Системи управління навчанням дозволяють науково педагогічним працівникам створювати автоматизовані опитування та завдання для здобувачів вищої освіти, які стають доступні онлайн. Більшість платформ підтримують різноманітні типи завдань, наприклад: тести; однорядкова/багаторядкова відповідь; варіант відповіді; замовлення; вільний текст; узгодження; есе; правда чи хибність/так чи ні; заповнити прогалини тощо.
Відгуки користувачів	Системи управління навчанням роблять можливим обмін відгуками як між здобувачами вищої освіти, так і між здобувачами вищої освіти та науково педагогічними працівниками. Науково-педагогічні працівники можуть створювати дискусійні групи, щоб надати здобувачам вищої освіти зворотній зв'язок, поділитися своїми знаннями і поліпшити взаємодію під час курсу [86]. Відгуки від здобувачів вищої освіти – це інструмент, який допомагає науково педагогічним працівникам покращити їхню роботу, допомагає визначити, що додати або видалити з курсу, і гарантує, що здобувачі

	вищої освіти почуватимуться комфортно та будуть максимально залучені до освітнього процесу.
Синхронне та асинхронне навчання	Здобувачі вищої освіти можуть навчатися як асинхронно (самостійно, тоді коли вони мають вільний час) за допомогою контенту курсу, такого як попередньо записані відео, PDF-файли, дискусійні форуми, так і синхронно, використовуючи відеоконференції, живі дискусії та чати безпосередньо з науково педагогічним працівником [87].
Аналіз процесу навчання	Системи керування навчанням включають інформаційні панелі для відстеження прогресу здобувачів вищої освіти [85]. Використання панелей допомагають співробітникам закладів вищої освіти завчасно виявляти проблемні моменти та приймати відповідні рішення для їх врегулювання.

Використання систем керування навчанням для закладу вищої освіти має відповідні переваги (таблиця 1.8) та недоліки [84].

Таблиця 1.9.

Переваги систем керування навчанням для ЗВО

Сумісність	Стандартизація форматів обміну інформацією у системах керування навчанням дозволяє обмін інформацією з однієї системи на іншу [85]
Доступність	Повноцінний інтерфейс, який використовується в системах керування навчанням, надає здобувачам вищої освіти з обмеженими можливостями змогу отримати доступ до всіх необхідних для вивчення обраного курсу матеріалів

Можливість повторного використання	Здатність системи керування навчанням повторно використовувати освітній контент, що було розроблено раніше. Критичний аспект у зниженні високих витрат на розвиток освітнього досвіду в умовах електронного навчання.
Можливість обслуговування	Системи керування навчанням дозволяють розробникам постійно вдосконалювати своє програмне забезпечення та краще адаптувати його до конкретної бази користувачів [87]
Адаптивність	Системи керування навчанням постійно вдосконалюються, оновлюються та швидко пристосовуються до використання нових моделей

Недоліки систем керування навчанням [89]:

– готовність з боку науково-педагогічних працівників вносити необхідні зміни у свої навчальні програми;

– готовність з боку науково-педагогічних працівників змінювати формат проведення занять з очних лекцій на онлайн-лекції.

– готовність з боку науково-педагогічних працівників використовувати наявні допоміжні матеріали в онлайн-курсах.

1.7. Використання методів і технологій інтелектуального аналізу даних у сучасних віртуальних освітніх середовищах

Інтелектуальний аналіз даних в освіті (англ. Educational data mining) – це галузь досліджень, пов’язана із застосуванням методів інтелектуального аналізу даних, машинного навчання та статистики до інформації, отриманої з закладів освіти (наприклад, університетів та інтелектуальних систем навчання)[107]. Мета інтелектуального аналізу даних в освіті — розробка, впровадження та вдосконалення методів

дослідження освітньої інформації. В залежності від освітнього закладу ці методи розділяють за кількома рівнями ієрархії для дослідження навчального прогресу студентів у відповідному контексті. Сфера інтелектуального аналізу даних в освіті тісно пов’язана зі сферою аналітики навчання. Отже ці дві сфери дуже часто порівнюються.

Інтелектуальний аналіз даних в освіті включає методи, інструменти і засоби дослідження, призначені для автоматичного вилучення значень з великих сховищ інформації, які створені або пов’язані з навчальною діяльністю осіб в освітніх установах. Досить часто обсяг цієї інформації є дуже великим (база знань розміром більше 100 Гб), а сама інформація досить деталізована та точна [108]. Наприклад, декілька систем керування навчанням відстежують інформацію за наступними показниками:

- у який час кожен здобувач вищої освіти отримував доступ до кожного навчального об’єкта;
- скільки разів кожен здобувач вищої освіти звертався до навчального об’єкта;
- скільки хвилин навчальний об’єкт відображався на екрані комп’ютера користувача.

Системи інтелектуального навчання записують дані кожного разу, коли студент надсилає рішення проблеми.

Системи інтелектуального навчання зберігають час подання, аналізують надіслане рішення на відповідність очікуваному рішенню, кількість часу, що минув із моменту останнього подання, порядок, у якому компоненти рішення були введені в інтерфейс. Точність отриманих даних є достатньою для проведення повноцінного аналізу: короткий сеанс із комп’ютерним навчальним середовищем (до 30 хвилин) створює необхідну кількість інформації для процесу аналізу.

Інтелектуальний аналіз даних також використовується у випадках, коли є необхідність у проведенні аналізу інформації з віртуальних навчальних середовищ від декількох закладів вищої освіти [109]. Наприклад: університетська довідка здобувача вищої освіти може містити наступну інформацію:

- тимчасово впорядкований перелік навчальних курсів, які здобувач вищої освіти прослухав;
- оцінку, отриману здобувачем вищої освіти за кожен навчальний предмет;
- час, коли здобувач вищої освіти обирав або змінював свою академічну спеціальність.

Інтелектуальний аналіз даних використовується для виявлення значущої інформації про різні категорії здобувачів вищої освіти і те, як вони навчаються, структуру предметних знань і вплив навчальних стратегій, вбудованих у різні навчальні середовища. Такий аналіз визначає інформацію, яку неможливо виявити методом спостереження [110]. Наприклад, аналіз даних із систем керування навчанням може виявити зв’язок між навчальними об’єктами, до яких студент мав доступ під час проходження курсу, та його остаточною оцінкою за курс.

Аналіз даних стенограми здобувача вищої освіти виявляє зв’язок між оцінкою здобувача вищої освіти за певний курс і його рішенням змінити академічну спеціальність. Така інформація надає уявлення про структуру навчального середовища, що дозволяє здобувачам вищої освіти, науково-педагогічним працівникам, адміністраторам і особам, які формують освітню політику закладу, приймати обґрунтовані рішення про те, як взаємодіяти з освітніми ресурсами, надавати доступ до них та керувати ними.

Головні цілі інтелектуального аналізу даних в освітньому середовищі наведено в таблиці 1.10.

Таблиця 1.10

Цілі інтелектуального аналізу даних

Прогнозування майбутньої поведінки здобувачів вищої освіти у навчанні	Досягається шляхом створення відповідних моделей навчання здобувачів вищої освіти, які включають характеристики здобувача вищої освіти, детальну інформацію про їхні знання, поведінку та мотивацію до навчання. Також вимірюється навчальний досвід здобувача вищої освіти та його загальна задоволеність навчанням.
Виявлення або вдосконалення моделей предметної області	За допомогою різних методів при використанні інтелектуального аналізу даних стає можливим відкриття нових і вдосконалення існуючих моделей. Приклади включають ілюстрування навчального контенту для залучення здобувачів вищої освіти і визначення оптимальної послідовності інструкцій для підтримки стилю навчання здобувача вищої освіти.

Вивчення ефектів освітньої підтримки	Пошук варіантів підтримки здобувачів вищої освіти, які можуть бути досягнуті за допомогою систем навчання.
Поглиблення наукових знань про навчання та здобувачів вищої освіти	Побудова та включення моделей поведінки здобувачів вищої освіти в області досліджень інтелектуального аналізу даних та використанні технологій і програмного забезпечення.

Виділяють наступні основні категорії зацікавлених сторін, залучених до інтелектуального аналізу освітніх даних таблиця 1.11.

Таблиця 1.11

Вимоги до категорій користувачів систем інтелектуального аналізу освітніх даних

Здобувачі вищої освіти	Використання досвіду та знань колишніх здобувачів вищої освіти, використовуючи інструменти інтелектуального аналізу даних в освіті. Для здобувачів вищої освіти інтелектуальний аналіз освітніх даних також використовується як інструмент для інформування про прогрес у навчанні. Також інтелектуальний аналіз дозволяє ефективно групувати здобувачів вищої освіти в онлайн-середовищі. Задача полягає в тому, щоб використовуючи складну для вивчення інформацію інтерпретувати її шляхом розробки дієвих моделей навчання [107].

Педагоги	Педагоги використовують засоби інтелектуального аналізу даних, для визначення варіантів організації та структуризації навчальної програми, методів подання інформації про курс та інструментів, що використовуватимуться для залучення здобувачів вищої освіти для досягнення оптимальних результатів навчання. Зокрема, дистиляція даних для техніки людського судження надає науково-педагогічним працівникам можливість отримати вигоду від інтелектуального аналізу даних, оскільки дозволяє науково-педагогічним працівникам швидко визначати моделі поведінки, які можуть підтримувати їхні методи навчання протягом курсу або покращити майбутні курси [108]. Науково-педагогічні працівники визначають індикатори, які показують задоволеність здобувачів вищої освіти, а також контролюють прогрес у навчанні.
Дослідники	Зосереджені на розробці та оцінці методів аналізу даних на предмет ефективності навчання. Широкий спектр областей для інтелектуального аналізу даних в освіті варіюється від використання аналізу даних для підвищення інституційної ефективності до результатів успішності студентів [109].
Адміністратори	Відповідають за розподіл ресурсів для реалізації в установах. Оскільки навчальні заклади все більше несуть відповідальність за якість та успішність здобувачів вищої освіти, адміністрування програм інтелектуального аналізу даних стає все більш поширеним у навчальних закладах. Науково-

	педагогічні працівники і консультанти стають більш активними у виявленні та вирішенні питань щодо здобувачів вищої освіти групи ризику.

Оскільки дослідження в галузі інтелектуального аналізу освітніх даних продовжує набирати обертів, безліч методів інтелектуального аналізу даних залучається до різноманітних освітніх контекстів. У кожному випадку метою є перетворення необробленої інформації про процес навчання у значущу, щоб мати змогу приймати кращі рішення щодо структури та траєкторії формування навчального середовища. Таким чином, інтелектуальний аналіз даних в освіті зазвичай складається з наступних етапів, представлених на рис 1.15.:

В інтелектуальному аналізі даних в освіті розрізняють два основних підходи [111]:

1. Дослідження з моделями. У цьому методі для розробки моделі використовуються наступні методи:

- прогнозування;
- кластеризації;
- інженерія знань людини.

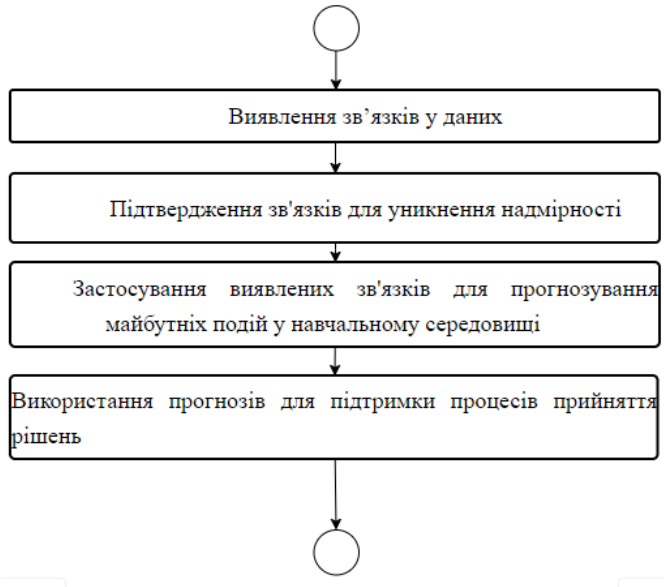


Рисунок 1.15. Етапи інтелектуального аналізу даних

На наступному етапі отримані результати використовуються як компоненти для іншого метода аналізу, а саме в прогнозуванні та аналізі зв’язків. При використанні методу прогнозування, передбачення створеної моделі використовується для прогнозування нової змінної. Для аналізу зв’язків створена модель дозволяє робити нові прогнози та вводити додаткові змінні в дослідження [110]. У багатьох випадках дослідження за допомогою моделей використовує перевірені моделі прогнозування, які підтвердили можливість узагальнення в різних контекстах. Ключова область застосування цього методу — виявлення взаємозв’язків між поведінкою здобувачів вищої освіти, характеристиками та контекстними змінними в навчальному середовищі.

2. **Обробка даних для людського судження.** Люди здатні робити висновки про дані, які виходять за рамки автоматизованого методу аналізу даних. Для використання інтелектуального аналізу даних в освіті інформація обробляється для людського судження для двох ключових цілей:

- ідентифікація;
- класифікація.

Для ідентифікації дані форматуються з урахуванням добре відомих закономірностей, які б в інакшому випадку залишилися неінтерпретованими. Наприклад, крива навчання, класична для педагогічних досліджень, є закономірністю, яка чітко відображає зв’язок між навчанням і досвідом у часі. Дані також обробляються з метою класифікації ознак даних, які для інтелектуального аналізу даних використовуються для підтримки розробки моделі прогнозування. Класифікація спрямована на оптимізацію процесу розробки моделі прогнозування. Метою цього методу є узагальнення та представлення інформації в корисний, інтерактивний та візуально привабливий для кінцевих користувачів спосіб, для інтерпретації великої кількості даних про процес навчання та підтримки прийняття рішень. Зокрема, цей метод

є корисним для науково-педагогічних працівників для розуміння інформації про використання матеріалів та ефективності цих матеріалів в межах навчального курсу.

Перелік основних застосувань інтелектуального аналізу даних в навчанні надано Крістобалем Ромеро та Себастьяном Вентурою [108]. У своїй таксономії вони виділяють відповідні сфери застосування інтелектуального аналізу даних в навчанні (рис 1.16).

Також до цієї таксономії входить створення навчального програмного забезпечення – EDM, що застосовується до систем керування навчанням, таких як Moodle. Moodle містить інформацію про користування системою:

- результати тестів;
- кількість завершених переглядів матеріалів курсу;
- участь у дискусійних форумах.

Рисунок 1.16. Сфери застосування інтелектуального аналізу даних

Інструменти інтелектуального аналізу даних використовуються для налаштування навчальної діяльності для кожного здобувача вищої освіти та адаптації темпу, з яким здобувач проходить та завершує курс.

Нові дослідження мобільного освітнього середовища також використовують інтелектуальний аналіз даних, для надання персоналізованого матеріалу мобільним користувачам, незважаючи на відмінності між мобільними пристроями та стандартними ПК і веббраузерами.

Нові додатки інтелектуального аналізу даних в освіті зосереджені на тому, щоб дозволити нетехнічним користувачам використовувати та брати участь у інструментах і діях інтелектуального аналізу даних, роблячи збір і обробку даних більш доступними для всіх користувачів систем інтелектуального аналізу даних в освіті. Приклади включають інструменти статистики та візуалізації, які аналізують соціальні мережі та їхній вплив на результати навчання та продуктивність.

1.8. Інтелектуальні навчальні системи

Інтелектуальна навчальна система (англ. ITS) — це комп'ютерна система, яка імітує науково-педагогічних працівників і спрямована на надання негайних і індивідуальних інструкцій або зворотного зв'язку здобувачам вищої освіти, без втручання науково-педагогічного працівника [109].

Головна мета інтелектуальної навчальної системи – забезпечення змістовного та ефективного навчання за допомогою різноманітних інформаційних технологій. Існує безліч прикладів використання інтелектуальних навчальних систем як у формальній освіті, так і в професійних умовах, де вони продемонстрували свої можливості та обмеження [109-110].

Існує тісний зв'язок між інтелектуальним навчанням, когнітивними теоріями навчання та дизайном, а також тривають дослідження для підвищення ефективності навчання. Мета інтелектуальних навчальних систем — відтворення продемонстрованих переваг індивідуального, персоналізованого репетиторства в контекстах, у яких здобувачі вищої

освіти мають доступ до інструкцій «один-до-багатьох» від одного науково педагогічного працівника (наприклад, лекції в аудиторії) або без науково педагогічного працівника взагалі. (наприклад, домашнє завдання онлайн). Мета розробки інтелектуальних навчальних систем: забезпечення доступу до високоякісної освіти для кожного здобувача вищої освіти.

Інтелектуальні навчальні системи будуються на основі однієї із чотирьох моделей:

1. **Доменна модель** (когнітивна модель або модель експертних знань) побудована на теорії навчання, такій як теорія ACT-R, яка намагається врахувати всі можливі кроки, необхідні для вирішення проблеми. Більш конкретно, ця модель «містить концепції, правила та стратегії розв'язання проблем у досліджуваній області».[109] Вона може виконувати кілька ролей:

- роль джерела експертних знань;
- роль стандарту для оцінювання успішності здобувача вищої освіти;
- роль еталону для виявлення помилок тощо.

Інший підхід до розробки моделей предметної області базується на теорії навчання на помилках продуктивності Стеллана Олссона [111], відомій як моделювання на основі обмежень (CBM)[31]. У цьому випадку модель предметної області представлена як набір обмежень на правильні рішення.

2. **Студентська модель** розглядається як надбудова над доменною моделлю. Це основний компонент інтелектуальної навчальної системи, який приділяє особливу увагу когнітивним й афективним станам здобувача вищої освіти та їх розвитку впродовж усього процесу навчання. Коли здобувач вищої освіти крок за кроком працює над процесом вирішення проблеми, інтелектуальна навчальна система бере участь у процесі, який називається трасуванням моделі. Кожного разу, коли модель здобувача вищої освіти відхиляється від моделі домену, система визначає або позначає, що сталася помилка. З іншого боку, модель здобувача вищої

освіти представляється у вигляді набору обмежень. Репетитори, засновані на обмеженнях, оцінюють рішення здобувача вищої освіти відповідно до набору обмежень і визначають чи були задоволені та порушені обмеження. Якщо є будь-які порушені обмеження, рішення здобувача вищої освіти визнається неправильним і інтелектуальна навчальна система надає зворотній зв'язок щодо цих обмежень. Репетитори, засновані на обмеженнях, надають негативний відгук (тобто відгук про помилки) або позитивний відгук.

3. Модель навчання приймає інформацію з предметної області та моделей здобувачів вищої освіти і робить вибір щодо стратегій навчання. У будь-який момент процесу вирішення проблеми здобувач вищої освіти має доступ до функції "поставити запитання" щодо подальших кроків відносно свого поточного розташування в моделі. Система розпізнає момент, коли здобувач вищої освіти відхиляється від правил моделі, і забезпечує своєчасний зворотний зв'язок для здобувача вищої освіти, що призводить до скорочення періоду часу для досягнення останнім необхідного рівня кваліфікації [112]. Модель науково-педагогічного працівника містить кілька сотень правил, які існують в одному з двох станів, вивченому чи невивченому. Щоразу, коли здобувач вищої освіти успішно застосовує правило до проблеми, система оновлює оцінку ймовірності того, що студент вивчив це правило. Система продовжує навчати здобувачів вищої освіти виконувати вправи, які потребують ефективного застосування правила, доки ймовірність того, що правило було успішно вивчено, не досягне принаймні 95% ймовірності. *Модуль відстеження знань* відстежує прогрес здобувача вищої освіти від проблеми до проблеми та створює профіль його сильних і слабких сторін відповідно до цілей навчання.

Система когнітивного навчання, розроблена Джоном Андерсоном з Університету Карнегі-Меллона, представляє інформацію з відстеження знань у вигляді візуального графіка успішності здобувача вищої освіти по

кожній із досліджуваних навичок, пов’язаних із розв’язуванням задач з алгебри. Коли студент запитує підказку або позначається помилка, дані відстеження знань оновлюються в режимі реального часу[68].

4. Модель інтерфейсу користувача «об’єднує три типи інформації, необхідні для ведення діалогу: знання про шаблони інтерпретації (для розуміння мовця) і дії (для створення висловлювань) у діалогах; знання предметної області, необхідні для передачі змісту; знання, необхідні для передачі намірів» [102].

Незважаючи на розбіжності між архітектурами інтелектуальних навчальних систем, кожна з яких зосереджується на різних елементах,

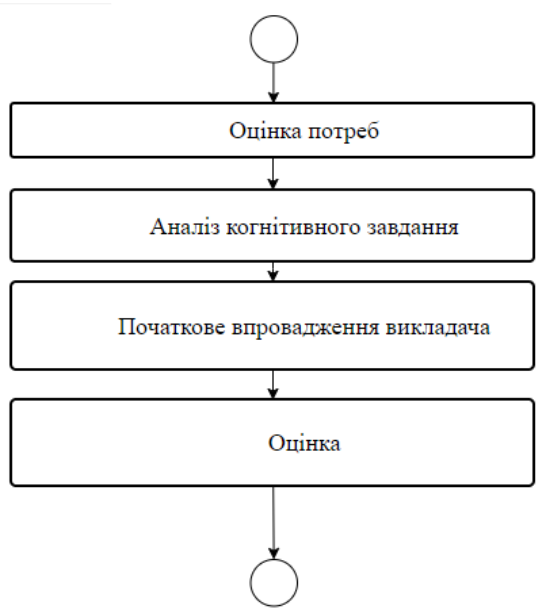


Рисунок 1.17. Ітераційні етапи розробки інтелектуальної навчальної системи

розробка інтелектуальної навчальної системи подібна до будь-якого процесу проектування навчання. Корбетт та іншими (1997) узагальнено дизайн і розробку інтелектуальної навчальної системи як такої, що складається з чотирьох ітераційних етапів (рис 1.17.)[102].

Перший етап, відомий як **оцінка потреб**, є загальним для будь-якого процесу проєктування, особливо розробки програмного забезпечення. Він передбачає аналіз діяльності здобувачів вищої освіти та консультації з експертами з предмету. Мета першого кроку —

визначення цілей навчання, окреслення загального плану навчальної програми та розроблення нової структури навчального плану, визначивши завдання в цілому та беручі до уваги можливу поведінку здобувачів вищої освіти при виконанні завдань. Оцінка успішності першого етапу проводиться за наступними ключовими показниками:

– ймовірність того, що здобувач вищої освіти зможе розв’язати поставлену задачу;

– час, потрібний для досягнення відповідного рівня успішності;

– ймовірність того, що здобувач вищої освіти буде активно використовувати ці знання в майбутньому;

- початкові характеристики науково педагогічних працівників і здобувачів вищої освіти, такі як попередні знання, оскільки обидві групи є користувачами системи.

Другий етап, відомий як, **когнітивний аналіз завдань**, є деталізованим підходом до програмування експертних систем з метою розробки дійсної обчислювальної моделі необхідних знань для вирішення проблем (рис. 1.18).

Рисунок 1.18. Основні методи розробки моделі предметної області

Хоча перший метод використовується найчастіше, експерти зазвичай не в змозі повідомити про когнітивні компоненти. Методи «думай вголос», за яких експертів просять повідомити вголос, про що вони думають під час вирішення типових завдань, можуть уникнути цієї

проблеми [102]. Спостереження за фактичною онлайн-взаємодією між науково-педагогічними працівниками та здобувачами вищої освіти надає інформацію, пов’язану з процесами, які використовуються під час вирішення проблем, що є корисним для побудови діалогу чи інтерактивності в системах навчання.

Третій етап, **початкове впровадження репетитора**, передбачає створення середовища для вирішення проблем, яке забезпечить підтримку автентичного процесу навчання. Наступним етапом є серія оцінних дій на завершальному етапі.

Четвертий етап, **оцінювання** включає**:**

- пілотні дослідження для підтвердження базової зручності та освітнього впливу;
- формувальні оцінки системи, що розробляється;
- параметричні дослідження, які перевіряють ефективність функцій системи;
- підсумкові оцінки кінцевого ефекту викладання: швидкість навчання та асимптотичні рівні досягнення.

Висновки до першого розділу

1. Події останніх десятиріч призвели до суттєвих змін в освітніх процесах як в Україні, так і в усьому світі. Інструмент, що забезпечує якісний та безперервний освітній процес, максимально незалежно від зовнішніх факторів впливу - це віртуальні освітні середовища.

2. Віртуальні освітні середовища мають відповідати актуальним формам організації освітнього процесу у закладах вищої освіти та надавати відповідні програмні інструменти.

3. Віртуальні освітні середовища мають бути побудованими на основі теоретичних та практичних підходів до побудови «систем керування взаємовідносинами».

4. Віртуальні освітні середовища мають включати засоби інтеграції

із зовнішніми системами підтримки різних аспектів освітнього процесу, такими як системи керування навчанням.

5. Віртуальні освітні середовища мають використовувати найсучасніші підходи до збільшення ефективності освітнього процесу, наприклад, таким як «Інтелектуальний аналіз даних в освіті».

РОЗДІЛ 2
ПРАКТИЧНІ РІШЕННЯ ЗАДАЧІ ПОБУДОВИ СУЧАСНОЇ ІАС УПРАВЛІННЯ ЗВО

2.1. Новітні архітектурні принципи та технологічні рішення

Перша версія «Інформаційно-аналітичної системи», що була розроблена та використовується в Херсонському державному університеті, архітектурно представляє собою звичайний «desktop»-застосунок[1]. Незважаючи на те, що функціональність системи охоплює великий спектр необхідних користувацьких функцій для підтримки навчального процесу закладу вищої освіти, застаріла архітектура не дозволяє достатньо ефективно адаптувати систему під мінливі вимоги сучасного світу. Наприклад: документообіг на кафедрах або факультетах.

Архітектура нової системи — розподілене клієнт-серверне середовище, яке складається з серверної частини, що реалізує усю бізнес-логіку, та клієнтської частини, яка складається із окремих застосунків, адаптованих під будь-який інтерфейс не залежно від пристрою, що використовує кінцевий користувач: здобувачі вищої освіти, науково-педагогічні працівники, науково-допоміжний персонал чи адміністратори [113]. Це архітектурне рішення в сукупності із сучасними технологіями, задіяними при розробці, дозволяє швидко реагувати на будь-які зміни у вимогах від закладу вищої освіти або Міністерства освіти і науки України.

Найсучасніший підхід до створення вебзастосунків - використання **інфраструктури програмних рішень** (англ. framework).

Інфраструктура програмних рішень (англ. Web Framework) або **каркас** вебзастосунків (англ. Web Application Framework) – це комплексний набір бібліотек, призначений для підтримки розробки складних програмних інтернет-застосунків [112-114]:

- інтернет-сервіси (англ. Web-services);
- інтернет-ресурси;

- прикладні програмні інтерфейси API (англ. Application Program Interface).

Інфраструктури програмних рішень забезпечують стандартизований спосіб створення та розгортання інтернет-застосунків у всесвітній павутині. Каркаси інтернет-застосунків, спрямовані на автоматизацію накладних витрат, пов'язаних зі звичайними діями, які виконуються під час розробки інтернет-застосунків. Наприклад, до складу каркасів інтернет-застосунків входять бібліотеки:

- бібліотеки для роботи із базами даних;
- бібліотеки шаблонів для типових рішень (наприклад: форма логіну на сайт);
- бібліотеки керування сесіями.

Головне призначення каркасів інтернет-застосунків — розробка динамічних вебсайтів.

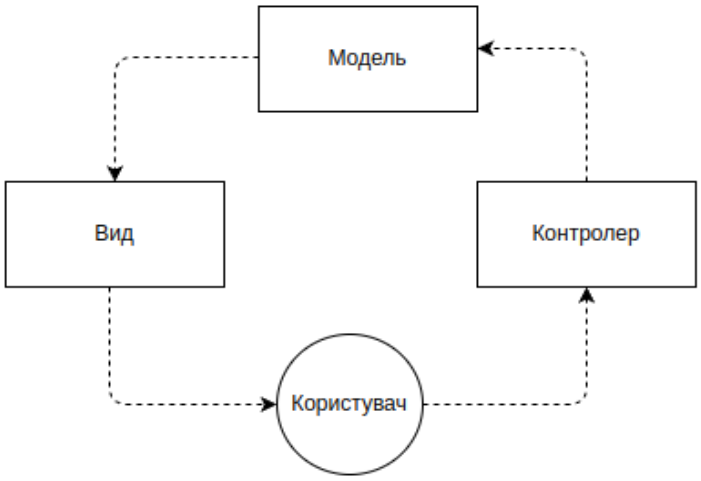


Рисунок 2.1. Архітектурний шаблон «модель-вид-контролер»

Каркаси інтернет-застосунків реалізовані на основі одного із трьох архітектурних шаблонів [115-116]:

1. Більшість каркасів інтернет-застосунків засновані на архітектурному шаблоні **модель-вид-контролер** (рис. 2.1), його головна архітектурна особливість — розділення на три складові частини:

- програмні модулі, що реалізують механізми маніпулювання даними — «модель»;
- програмні модулі, що реалізують бізнес-логіку — «контролер»;
- програмні модулі що реалізують інтерфейс користувача — «вид» (рис.2.1).

Така архітектура надає наступні переваги для розробників:

- оптимізація процесів розробки та внесення змін до програмного коду застосунка на відповідних етапах життєвого циклу, шляхом розділення програмного коду на логічні частини;
- повторне використання коду;
- застосування розмаїття інтерфейсів.

2. «Архітектура на основі дій» проти «архітектури на основі залучення». Каркаси інтернет-застосунків архітектури модель-вид-контролер дотримуються архітектурних шаблонів, заснованих на принципі «архітектура на основі дій» (англ. «Push-based»). У каркасах інтернет-застосунків термін «дія» означає процеси, які виконують необхідну обробку даних, після чого передають інформацію на рівень перегляду для відображення результатів [113].

Альтернативою «архітектури на основі дій» є «архітектура на основі залучення», інша назва якої «компонентна архітектура». Процес розробки з використанням каркасів інтернет-застосунків «компонентної архітектури» починається з рівня відображення, до якого під’єднуються методи з кількох контролерів за потреби.

3. Трирівнева структура. Архітектура каркасів інтернет-застосунків розділяється на три рівня:

- клієнт — веббраузер, відповідає за обробку HTML-сторінок, згенерованих прикладним рівнем;
- застосунок — відповідає за бізнес-логіку, взаємодіє з клієнтом за допомогою протоколу HTTP;
- база даних — відповідає за зберігання та обробку інформації.

Архітектурний шаблон, який використовується при розробці системи «KSU24» — це модель-вид-контролер (Model-View-Controller).

Таблиця 2.1

Задачі компонент шаблону модель-вид-представлення

Модель	Центральний компонент каркасу інтернет-застосунків, функції якого: - управління усією інформацією застосунка, - обробка запитів від контролера. Це класи, що представляють собою динамічну структуру даних застосунка, яка не залежить від інтерфейсу користувача. Задача моделі - це керування інформацією, правилами та логікою застосунка. У сучасних каркасів інтернет-застосунків клас однієї моделі представляє одну таблицю з бази даних (виключення: таблиці, які відповідають за організацію типу зв'язку між таблицями "багато-до-багатьох" (англ. «many-to-many») та «технічні» таблиці створені системою керування базою даних (наприклад: індекси)) [114-115].
Вид	Компонент відповідальний за відображення інформації у вигляді звичайних сторінок, графіків, діаграм, таблиць та ін. Каркаси інтернет-застосунків передбачають створення кількох видів (відображень) для одного джерела інформації, наприклад: представлення фінансового звіту за певний період у вигляді гістограми для керівництва та у табличному вигляді для бухгалтерів. Наприклад у каркасі інтернет-застосунків Django [114] роль представлення виконують сторінки-шаблони HTML, які визначають інтерфейс користувачів у браузері, а не представляють безпосередньо

	окремий застосунок інтерфейсу користувача. У зв'язку з цим Django вважає за краще називати цей вид об'єктів «шаблоном» (англ. «template»). У Django представлення/шаблон використовуються контролером/представленням у процесі підготовки відповіді клієнту.
Контролер	Компонент відповідальний за отримання вхідних даних та перетворення їх на відповідні команди моделей чи видів. Головна функція контролера — обробка подій користувацького введення, такі як натискання кнопок або рух миші. Особливості екземплярів класів контролерів: - у будь-який момент часу кожен екземпляр класу контролера пов'язаний з одним екземпляром класу представлення та одним екземпляром класу моделі, хоча один екземпляр класу моделі може взаємодіяти із багатьма екземплярами класів контролерів; - тільки один екземпляр класу контролера є «активним» в певний момент часу, в який він отримує дані користувача; глобальний об'єкт диспетчера вікон відповідає за налаштування поточного активного контролера; - якщо команди від користувача спонукають до змін у класі моделі, контролер сигналізує про необхідність такої зміни, проте саме клас моделі відповідає за внесення цих змін у відповідні вже існуючі екземпляри класу моделі [115-116]. У каркасі інтернет-застосунків Django об'єкт, що відповідає за функції контролера носить назву, «представлення» (вид) замість «контролер». Представлення Django — це функція, яка отримує вебзапит і повертає вебвідповідь [116].

Структурно компонент «модель» складається з наступних частин:

- частина, яка реалізує частку бізнес-логіки вебзастосунку (цю частину частіше за все і називають просто моделлю), що відповідає за процес перевірки (валідації) даних перед їх відправкою у базу даних;
- частина, яка формує, надсилає запити до системи керування базами даних, та отримує відповідну інформацію у необхідному вигляді, перетворюючи її у відповідний формат, називається об'єктно-реляційне відображення (англ. Object–Relational Mapping) [114].

У теорії комп'ютерних наук об'єктно-реляційне відображення — це техніка програмування призначена для організації процесу обміну даними між реляційною базою даних і сукупністю об'єктів об'єктно-орієнтованої мови програмування [115]. Головна особливість об'єктно-реляційного відображення — створення копії структури таблиць бази даних в оперативній пам'яті у вигляді віртуальних об'єктів, яка використовується іншими частинами застосунку. В об'єктно-орієнтованому програмуванні завдання керування даними стосуються інформації отриманої із таблиць бази даних, поєднуючи скалярні значення в об'єкти. Навпаки, реляційні бази даних, такі як SQL, групують скаляри в кортежі, які потім записуються в таблиці. Кортежі й об'єкти — засіб групування значень у вигляді іменованих полів, для забезпечення механізмів керування колекцією записів із бази даних як єдиною складеною сутністю.

Задачі кортежів та об'єктів моделей:

- керування життєвим циклом (задачі створення, редагування та видалення рядків у самій базі даних, та за необхідності звернення до компонента «прибиральник сміття» (англ. «garbage collector»));
- керування зв'язками з іншими сутностями (мається на увазі саме посилання на об'єкти, а не посилання на зовнішній ключ відповідної таблиці);
- спадкування (не існує в реляційних базах даних).

Модель відповідає за керування сукупністю об'єктів як одним цілим в одному обчислювальному процесі, тоді як для керування кортежами

бази даних задіються механізми синхронізації: «блокування», «злиття» та «повторні спроби». Об’єктно-реляційне відображення забезпечує автоматизовану підтримку відображення кортежів бази даних в об’єкти моделей і навпаки [116].

Головна задача системи об’єктно-реляційного відображення — приведення логічного представлення об’єктів до канонічної форми, яка зберігається в базі даних, враховуючи властивості об’єктів та зв’язки між ними, для забезпечення функцій перезавантаження та оновлення. За наявності функцій зберігання та пошуку, об’єкти вважаються *постійними*.

Архітектура вебзастосунків будується відповідно до архітектурних правил браузерів і протоколів, таких як HTTP. Задачі генерації та обслуговування вебсторінок забезпечуються сервером. Подальша зміна сторінок забезпечується одним із наступних підходів:

- **без відправки запиту на сервер**, забезпечується безпосередньо браузером та технологією JavaScript. Зміни вмісту сторінок на стороні клієнта дозволяють робити оновлення окремих частин сторінки, що для кінцевого користувача створює досвід роботи, подібний до звичайного настільного застосунка, але з обмеженнями, пов’язаними із рушієм JavaScript і працювати у браузері користувача.

- **з відправкою запиту на сервер**. Для зміни вмісту сторінки на стороні сервера необхідне повне оновлення сторінки, що дозволяє використання будь-якої мови програмування та використання більшої обчислювальної потужності.

В сучасних інтернет-застосунках, використовується поєднання цих двох підходів. Інтернет-застосунки, які активно використовують мову JavaScript для оновлення окремих частин сторінок, називаються односторінковими застосунками[117].

Односторінковий застосунок (Single Page Application, SPA) [188] — це інтернет-застосунок, взаємодія із користувачами у якому побудована на динамічному переписуванні частини поточної вебсторінки новою

інформацією з сервера, без необхідності повного перезавантаження сторінки. Головна відмінність односторінкових застосунків — в односторінковому застосунку сторінки ніколи не оновлюються (мається на увазі процес повного завантаження сторінки із сервера з новою інформацію у відповідь на дію користувача), замість цього уся необхідна інформація, а саме HTML сторінки, сценарії JavaScript та стилі CSS або отримується браузером в процесі першого завантаження сторінки [118]. У відповідь на дії користувача необхідна інформація динамічно завантажується та додається у відповідне місце на сторінці за необхідності. Головний інструмент односторінкового застосунку — мова програмування JavaScript, яка виконується у веббраузері для відображення інтерфейсу користувача (UI) та відповідає за зв'язок із вебсервером. Найбільш розповсюджені бібліотеки та каркаси інтернет-застосунків, для створення односторінкових вебзастосунків та їх особливості наведено в таблиці 2.2.

Таблиця 2.2.

Технології розробки односторінкових вебзастосунків

AngularJS	Шаблони AngularJS [127] базуються на двосторонньому зв'язуванні даних інтерфейсу користувача. Зв'язування даних — це спосіб автоматичного оновлення сторінок при зміні моделей, а також оновлення моделей при зміні сторінок. Шаблон HTML компілюється в браузері. На етапі компіляції створюється HTML документ, який повторно відображається вебпереглядачем в режимі реального часу. Цей процес повторюється при подальших зверненнях до сторінок. У традиційному програмуванні HTML генерується на стороні сервера, а такі компоненти, як контролер і модель, взаємодіють у серверному середовищі для створення нових представлень HTML. У каркасі інтернет-застосунків

	AngularJS стани контролера та моделі зберігаються в браузері клієнта. Таким чином, нові сторінки генеруються без взаємодії з серверною частиною.
Ember.js	Каркас інтернет-застосунків для створення вебзастосунків на стороні клієнта, заснований на архітектурному шаблоні програмного забезпечення модель–вид–контролер. Він надає розробникам функції для створення масштабованих односторінкових застосунків, шляхом включення загальних ідіом і найкращих практик у структуру, яка забезпечує багату об'єктну модель, декларативне двостороннє зв'язування даних, обчисленні властивості, автоматичне оновлення шаблонів на основі Handlebars.js і маршрутизатор для керування станом застосунка.
ExtJS	Каркас інтернет-застосунків для створення вебзастосунків на стороні клієнта, який має: - власну систему подій; - власну систему керування вікнами та макетами; - власну систему керування станом; - різні компоненти інтерфейсу користувача; - власну систему класів з динамічним або статичним завантажувачем. Застосунок, створений за допомогою ExtJS, може існувати окремо (зі станом у браузері), або взаємодіючі із сервером (наприклад, з REST API) [124].
Meteor.js	Каркас інтернет-застосунків JavaScript із так званим «повним стеком» розробки (клієнт-сервер), створений для односторінкових програм, в якому реалізована оптимізований механізм прив'язки даних, у порівнянні із Angular, Ember або ReactJS. Він використовує протокол розподілених даних і

	шаблон публікації–підписки для автоматичного поширення змін даних клієнтам у режимі реального часу, не вимагаючи від розробника реалізації коду для синхронізації [121-123].
React	Бібліотека JavaScript призначена для створення інтерфейсів користувача [119-130]. Вона підтримується таким відомими соціальними мережами як Facebook та Instagram, а також спільнотою окремих розробників і корпорацій. React використовує розширення синтаксису для JavaScript під назвою JSX, яке є сумішшю JS і підмножини HTML. React постачається разом із бібліотекою Redux, яка додає можливості керування станами.

Головною технологічною складовою програмних бібліотек наведених в таблиці 2.2. є технологія AJAX **(**«**A**synchronous **J**avaScript and **X**ML»**)** — підхід до побудови користувацьких інтерфейсів вебзастосунків, за яких вебсторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і власноруч довантажує необхідні користувачу дані. AJAX — один з компонентів концепції динамічного HTML. За допомогою AJAX вебзастосунки надсилають та отримують дані з сервера асинхронно (у фоновому режимі), не втручаючись у відображення та поведінку існуючої сторінки. Відокремлюючи процес обміну інформацією від процесу її відображення, AJAX дозволяє вебсторінкам і, за розширенням, вебзастосункам змінювати вміст сторінок динамічно, без необхідності у оновлення всієї сторінки [132]. Сучасні реалізації використовують технологію JSON замість XML. В літературі AJAX розглядається не як конкретна технологія, а у вигляді концепції програмування. Зміст вебсторінки змінюється засобами технології JavaScript для динамічного відображення — для реалізації взаємодії користувача з оновленою інформацією. Вбудований об’єкт XMLHttpRequest використовується для виконання сценаріїв AJAX на

вебсторінках, дозволяючи вебзастосункам завантажувати вміст без оновлення сторінки.

У якості формату для обміну даними між клієнтською та серверною частинами інтернет-застосунків використовується формат JSON — стандартний відкритий формат файлів та формат, що використовується для обміну інформацією, завдяки застосуванню зрозумілого для людини тексту для збереження та передачі об’єктів, які виглядають як пари атрибут–значення та масивів (або інших значень, що серіалізуються).

Формат був створений як похідний від JavaScript, проте велика кількість сучасних мов програмування мають необхідні інструменти для створення та аналізу даних у цьому форматі [133]. JSON-файли мають розширення .json (табл. 2.3).

Таблиця 2.3.

Формати даних, які підтримуються форматом JSON

Числовий	Десяткове число, що має знак, включає дробову частину, що використовує експоненціальний запис, але не може містити нечислові значення, наприклад «NaN». У мові JavaScript використовується числовий формат з подвійною точністю з плавучою крапкою IEEE-754 для будь-яких власних числових значень [130].
Рядки	Послідовність із нуля або більше символів Unicode. Рядки розділені подвійними лапками та підтримують синтаксис екранування зворотньої косої риски.
Логічні значення	Будь-яке зі значень true або false.
Масив	Упорядкований перелік, що містить нуль або більше елементів, незалежно від їх типу даних. Для масивів використовується нотація в квадратних дужках з елементами, розділеними комами [132].

Об’єкт	Набір пар «ім’я» – «значення», у якому «ім’я», яке також називають «ключ» представляє собою рядок [131-133]. Поточний стандарт ECMA стверджує: «Синтаксис JSON не накладає жодних обмежень на рядки, які використовуються як імена, не вимагає, щоб рядки імен були унікальними, і не надає жодного значення впорядкуванню пар ім’я/значення». Об’єкти розділені фігурними дужками та використовуються коми для розділення кожної пари, тоді як у кожній парі двокрапка «:» застосовується для відокремлення ключів або імен від їх значень.
null	Порожнє значення.

Числа в форматі JSON не залежать від їх представлення в різних мовах програмування. Формат дозволяє серіалізацію чисел довільної точності, що у певних випадках призводить до проблем з переносимістю [132]. Наприклад: оскільки не проводиться розрізнення між цілими числами та значеннями з плавучою крапкою, у деяких реалізаціях числа 42, 42.0 і 4.2E+1 можуть розглядатися, як одне й те саме число, а в інших – ні. У стандарті JSON немає внутрішньої втрати точності при серіалізації двійкового представлення числа з плавучою комою на машинному рівні (наприклад, binary64) у зрозуміле людині десяткове представлення (як числа в JSON) і назад, оскільки існують опубліковані алгоритми, щоб зробити це точно і оптимально.

Хоча формат JSON забезпечує синтаксичну структуру для обміну даними, однозначний обмін даними також вимагає згоди між розробником і користувачем щодо семантики конкретного використання синтаксису JSON. Одним із прикладів необхідності такої угоди є серіалізація типів даних, визначених синтаксисом JavaScript, які не є частиною стандарту JSON, наприклад, дата, функція, регулярний вираз і спеціальне значення undefined. Формат JSON використовується наступними технологіями:

– «JSON-RPC» — це протокол, що використовується для віддаленого виклику процедур (англ. «Remote Procedure Call»), та заснований на форматі «JSON» для заміни застарілих протоколів («XML-RPC» та «SOAP»). Протокол «JSON-RPC» надає системі функціональність для надсилання сповіщень (інформація, що відправляється на сервер та не потребує відповіді) і численні запити на сервер, який оброблює їх та надає відповідь у будь-якому порядку.

– «Асинхронний JavaScript і JSON» (скорочено «AJAJ») відноситься до тієї ж методології розробки динамічних вебсторінок, як і технологія «Ajax», але замість «XML» форматом даних є «JSON». «AJAJ» — це техніка розробки вебзастосунків, яка забезпечує вебсторінкам можливість запитувати нові дані після їх завантаження у веббраузер. Зазвичай веббраузер обробляє нові дані з сервера після того як користувач виконає на цій вебсторінці певну дію. Наприклад, інформація яка вводиться користувачем у вікні пошуку, потім надсилається клієнтським кодом на сервер, який негайно відповідає випадаючому списку відповідних елементів з бази даних [131].

– Деякі реляційні бази даних, такі як PostgreSQL і MySQL, додали формат JSON у якості стандартного типу даних. Це дозволяє розробникам зберігати дані JSON безпосередньо в реляційній базі даних без необхідності конвертувати їх в інший формат даних.

Разом із каркасами інтернет-застосунків для оптимізації процесу відображення великої кількості інформації, що надходить зі сторони сервера, використовують так звану технологію серверного рендерингу (англ. Server-side rendering). Це технологія, яка використовується у веброзробці для відтворення вебсторінок на сервері, а не на стороні клієнта (тобто, у веббраузері користувача). Цей підхід передбачає створення коду HTML, CSS і JavaScript на сервері та надсилання його клієнту як повністю сформовану вебсторінку, а не використання відтворення на стороні клієнта для динамічного створення вмісту в

браузері [132]. Переваги використання серверного рендерингу наведено в таблиці 2.4 [157].

Таблиця 2.4.

Технічні переваги технології серверного рендерингу

Оптимальний час першого завантаження	За допомогою серверного рендеренга - сервер надсилає повністю сформовану вебсторінку клієнту, що значно скорочує час, потрібний користувачеві для перегляду початкового вмісту.
Оптимізація для пошукових систем	Оскільки пошукові системи покладаються на вміст HTML для індексування вебсторінок, серверний рендеринг покращує SEO-оптимізацію вебсайту, шляхом оптимізації процесів сканування та індексації вмісту пошуковими системами.
Оптимізована продуктивність на повільних пристроях	Серверний рендеринг підвищує продуктивність, переклавши роботу з відображення на сервер для оптимізації роботи на пристроях із повільним процесором або обмеженою пам’яттю.
Покращена безпека	Серверний рендеринг призначено для покращення безпеки вебсайтів шляхом зменшення ризику атак «міжсайтових сценаріїв» (англ. Cross Site Scripting “XSS”).

Технологія серверного рендерингу реалізована для різних мов програмування та фреймворків, включаючи Node.js та Ruby on Rails. Популярні каркаси інтернет-застосунків, у яких упроваджено технологію серверного рендерингу: Next.js, Nuxt.js і Angular Universal.

Недолік технології серверного рендерингу — складність налаштування та обслуговування середовища на сервері [158]. Крім того,

серверний рендеринг збільшує навантаження на сервер, оскільки задача сервера — генеравація HTML для кожного запиту.

В проєкті «KSU24» використовується технологія серверного рендерингу **Node.js**. - це кросплатформне середовище виконання JavaScript з відкритим кодом, яке призначене для роботи в операційних системах Windows, Linux, Unix, macOS тощо. Node.js використовує механізм «JavaScript V8» та дозволяє виконання інструкцій JavaScript поза веббраузером.

Технологія Node.js надає розробникам можливості використання мови програмування JavaScript для програмування інструкцій командного рядка сервера та виконання сценаріїв безпосередньо на стороні сервера. Метод виконання коду JavaScript на стороні сервера використовується для створення та обробки вмісту вебсторінки динамічно перед надсиланням сторінки у веббраузер користувача. Node.js реалізує сучасну парадигму «JavaScript скрізь», яка передбачає використання однієї мови програмування для повного циклу розробки вебзастосунків, у протилежність до застосування однієї або більше мов програмування для сценаріїв на стороні сервера та іншого набору мов програмування на стороні клієнта.

Технологія Node.js побудована на керованій подіями архітектурі з можливістю асинхронного введення-виведення, яка спрямована на оптимізацію пропускної здатності та масштабованості черги запитів у вебзастосунках із багатьма операціями введення/виведення, а також для вебзастосунків у реальному часі (наприклад, програм для спілкування в реальному часі та браузерних ігор) [156].

Node.js дозволяє створення вебсерверів та інструментів для роботи із комп’ютерними мережами за допомогою JavaScript і колекції «модулів», які реалізують базові функції:

- модулі для роботи із файлами та папками;

- модулі для роботи із мереживими протоколами («DNS», «HTTP», «TCP», «TLS/SSL» або «UDP»);
- модулі для роботи зі сховищами двійкової інформації (буферів);
- модулі для роботи з криптографічними алгоритмами;
- модулі для обробки «потоків даних».

Елементи «Node.js» використовують асинхронний програмний інтерфейс (англ. «API»), що застосовується для зменшення складності написання серверних програм.

Node.js призначено для створення мережевих програм, таких як вебсервери. Найсуттєвіша відмінність між технологією Node.js і, наприклад, мовою серверних сценаріїв PHP — переважна кількість функцій у PHP блокуються до завершення (наступна команда може почати виконання лише після завершення виконання попередньої команди), у той час як в Node.js виконання функцій не блокуються (велика кількість команд виконується одночасно або паралельно і використовує зворотні виклики для надсилання сигналів про завершення або невдачу).

Архітектура платформи Node.js дозволяє розробку швидких вебсерверів мовою JavaScript, використовуючи технології “програмування, керованого подіями”. Надає розробникам функції для створення масштабованих серверів без використання потоків за допомогою спрощеної моделі, яка дозволяє використання “зворотніх викликів” для надсилання сигналів щодо завершення завдання [157].

З технічної сторони Node.js — це середовище виконання JavaScript, яке обробляє вхідні запити в циклі, який називається циклом подій (event loop).

Технічні особливості платформи Node.js, наведені в таблиці 2.5.

Таблиця 2.5

Технічні особливості платформи Node.js

Цикл подій Node.js має однопотокову архітектуру	Цикл подій у Node.js заснований на неблокуючих викликах введення-виведення, що забезпечує підтримку десятків тисяч одночасних з'єднань без витрат на перемикання контексту потоку. Конструкція спільного використання одного потоку серед усіх запитів, які використовують шаблон «спостерігач», призначена для створення високорозподілених програм, у яких будь-яка функція, яка призначена для виконання введення-виведення, використує зворотній виклик. Для підтримки однопотокового циклу подій, Node.js використовує бібліотеку «libuv», яка, зі свого боку, використовує фіксований набір потоків для обробки неблокуючих асинхронних операцій введення-виведення. Черга потоків керує виконанням паралельних завдань у Node.js. Виклик функції основного потоку розміщує завдання в спільній черзі завдань [157]. За своєю суттю «неблокуючі системні функції», (наприклад, мережеві), перетворюються на «неблокуючі сокети» на стороні ядра, тоді як за своєю суттю «блокуючі системні функції», наприклад, введення/виведення файлів, виконуються із блокуванням у власних потоках виконання. Після того як один потік з набору потоків починає виконання завдання, він повідомляє про це керуючий потік, який, у свою чергу, «прокидається» та виконує зареєстрований зворотній виклик. Така однопотокова архітектура має серйозний недолік — Node.js не дозволяє вертикальне масштабування шляхом збільшення кількості ядер

	процесора машини, на якій він працює, без використання додаткового модуля, такого як «кластер», «StrongLoop Process Manager» або «pm2». Проте використання модуля libuv надає розробникам можливість збільшувати стандартну кількість потоків у загальному переліку потоків. Серверна операційна система (ОС) розподілить ці потоки між кількома ядрами [157 – 158]. Додаткова проблема — довготривалі обчислення та інші завдання, які пов'язані з процесором, здатні призупинити весь повний набір процесів до того моменту, як «складний процес» буде завершено.
Застосунок керування пакетами npm	Менеджер пакетів для серверної платформи Node.js. Відповідає за процес встановлення програм Node.js з реєстру npm, організовуючи встановлення та керування сторонніми програмами Node.js.
Цикл подій	Node.js реєструється в операційній системі для доступу до механізмів повідомлень про асинхронні події введення-виведення, такі як нові підключення. У середовищі виконання Node.js події викликають зворотні виклики, і кожне з'єднання обробляється як невелика група команд. У Node.js цикл подій використовується для організації одночасного введення-виведення [156]. Інші вебсервери, які керуються подіями, не дозволяють неявні виклики процесів, що є можливим у циклі подій в Node.js. Натомість система назначає зворотні виклики для автоматичного введення сервера у цикл подій. Після того, як зворотній виклик буде визвано, Node.js вийде із циклу подій, коли немає подальших зворотніх викликів для виконання.

«Нативне» зв’язування.	Node.js надає можливість створювати «надбудови» за допомогою API на основі C під назвою N-API, які можуть бути використані для створення завантажуваних (імпортованих) модулів .node із вихідного коду, написаного на C/C++. Модулі безпосередньо завантажуються в пам’ять і виконуються в середовищі JS як елементарні модулі CommonJS. Реалізація Node-API покладається на внутрішні C/C++ Node.js і об’єкти V8, які вимагають від користувачів імпортувати (#include) специфічні заголовки Node.js у свій вихідний код [158]. Незважаючи на те, що основні функції Node.js містяться у вбудованій бібліотеці JavaScript, модулі, написані на C++, можна використовувати для розширення можливостей і підвищення продуктивності програм.

2.2. Перевірка якості програмного забезпечення в процесі побудови сучасної інформаційно-аналітичної системи керування ЗВО

Тестування програмного забезпечення — це перевірка складових частин і поведінки тестованого програмного забезпечення шляхом верифікації та підтвердження [148-149]. Тестування програмного забезпечення також надає незалежний, об’єктивний погляд щодо програмного забезпечення в цілому, для надання компанія можливостей надання оцінок та передбачення ризиків пов’язаних із упровадженням та використанням певного програмного засобу. Етап тестування є дуже важливим для розробки та ефективного впровадження інформаційно-аналітичної системи керування закладом вищої освіти, оскільки згідно до гнучкої (англ. Agile) моделі розробки, нові вимоги до окремих модулів системи «KSU24», і до системи в цілому дороблюються та змінюються

протягом навчального року. Такі зміни розподіляються на дві категорії [150]:

- **внутрішні вимоги** надходять безпосередньо від «прямих» користувачів системи: науково-педагогічних працівників та адміністрації Херсонського державного університету;

- **зовнішні вимоги** формуються у результаті реагування на зміни чинного законодавства щодо освіти, правил оформлення силабусів навчальних дисциплін тощо;

Низку методів тестування, що використовуються у проєкті «KSU24», зазначено в таблиці 2.6.

Таблиця 2.6.

Методи тестування

Аналіз вимог	Перевірка вимог щодо повноти та правильності програмного застосунка в різних контекстах: - галузеві перспективи; - бізнес-перспективи; - доцільність і життєздатність упровадження; - зручність використання; - продуктивність; - безпека; - інфраструктурні міркування [148].
Аналіз архітектури	Перегляд архітектури програмного застосунка та загального дизайну продукту.
Робота з розробниками застосунків	Вдосконалення методів написання програмного коду, впровадження шаблонів проєктування, тестів, які можуть бути реалізовані у вигляді частини програмного коду на основі відповідних методів (наприклад, метод граничних умов) [150].

Виконання	Використання застосунку з метою вивчення поведінки;
Розгортання	Перегляд інфраструктури розгортання та пов’язаних сценаріїв і автоматизації [149].
Моніторинг	Участь у виробничій діяльності за допомогою методів моніторингу та спостереження.

Задача тестування програмного забезпечення — надання користувачам повної незалежної інформації, якість програмного застосунка та можливі ризики невдалого його впровадження.

Тестування визначає те, наскільки «правильним» є програмний засіб за припущенням конкретних гіпотез, але все одно використання тестування, не забезпечує повного виявлення усіх помилок та збоїв у програмному застосунку. Замість цього тестування містить критику або проводить порівняння між станом і поведінкою застосунка із, так званими, тестовими «оракулами» — принципами або механізмами, які дозволяють розпізнавання потенційної проблеми [151]. Такі «оракули» включають (але не обмежуються ними):

- специфікації;
- контракти;
- програмні продукти, які наводяться у якості прикладу;
- минулі версії того самого продукту;
- висновки щодо наміченої чи очікуваної мети;
- очікування користувачів або клієнтів;
- відповідні стандарти;
- державні закони або специфічні вимоги.

Головна мета процесу тестування програмного засобу — це виявлення збоїв у програмному застосунку для визначення та виправлення дефектів. Ця мета підкреслюється наступним виразом: «Тестування не використовується для того, щоб встановити, чи продукт

функціонує належним чином за будь-яких умов, а лише те, що він не функціонує належним чином за певних умов» [152-153]. До тестування програмного забезпечення входять наступні процеси:

- перевірка програмного коду, із виконанням цього коду в різних програмних середовищах та за різних умов;
- перевірка відповідних критеріїв коду: чи виконує програмний код те, що зазначено у його специфікації. У сучасних підходах до організації процесу розробки програмного забезпечення різних рівнів складності, процес тестування може бути організованим повністю окремо від команди розробників. Як і команда розробників, члени команди тестувальників розподіляються за відповідними ролями. Інформація, яку команда отримує під час, або після процесу тестування програмного засобу, може використовуватись навіть для внесення змін у процес розробки програмного засобу.

Будь-який програмний засіб розробляється із урахуванням відповідної цільової аудиторії. Наприклад: аудиторія користувачів відеоігор кардинально відрізняється від аудиторії користувачів систем керування закладами вищої освіти. Тестування надає можливість можливість надання оцінок прийняття застосунка кінцевими користувачами, цільовою аудиторією, покупцями, інвесторами та іншим зацікавленими сторонами.

«Баги» (помилки, англ. bugs) у програмному застосунку виникають у результаті такого процесу: програміст робить помилку, що призводить до несправності (дефекту) у висхідному коді програмного забезпечення [154]. У випадку періодичності виникнення однакових несправностей, у певних ситуаціях система видають хибні результати, що може спричинити програмний збій.

Не всі несправності обов’язково призводять до збоїв. Наприклад, помилки в «мертвому» коді ніколи не призводять до збоїв. Помилка, яка

не виявила збоїв, може призвести до збою під час зміни середовища виконання. Приклади подібних змін у середовищах включають:

- програмні засоби, які переносяться на нові апаратні платформи комп'ютерів;
- програмні засоби, які взаємодіють з іншими версіями програмного забезпеченням, ніж ті, які були надані розробникам у якості прикладу;
- зміни у вихідних даних.

Також, не всі помилки програмного забезпечення викликані помилками програмістів [153-154]. Одним із найпоширеніших джерел найбільш «коштовних дефектів» є, так звані, «прогалини» у вимогах, тобто нерозпізнані вимоги користувачів, які призводять до неправильної реалізації цих вимог розробником програми. Прогалини у вимогах часто відносяться до нефункціональних вимог:

- масштабованість;
- зручність обслуговування;
- продуктивність;
- безпека.

Фундаментальна проблема тестування програмного забезпечення полягає в тому, що тестування за всіх комбінацій вхідних даних і попередніх умов (початкових станів) є повністю неможливим, навіть, у відносно невеликому програмному продукті. Це означає, що залежність кількості помилок у програмному забезпеченні не може бути визначена, беручі до уваги масштаби програмного продукту, а ті дефекти, що рідко виникають, не виявляються шляхом «ручного» тестування та впровадження. Більш важливо те, що нефункціональні параметри якості — «зручність використання», «масштабованість», «продуктивність», «сумісність і надійність» є дуже суб'єктивними, оскільки характеристика, яка є важливою, з точки зору одного користувача, може виявитись цілком неприпустимою для іншого.

Розробники програмних засобів не мають можливостей для перевірки усіх існуючих варіантів використання для усіх можливих апаратних та програмних середовищ, проте використання комбінаторного дизайну тестів для визначення мінімально-необхідної кількості тестів дозволяє отримати достатньо повне покриття функціональності застосунка [155]. Комбінаторний дизайн тестів забезпечує максимально широке тестове покриття за допомогою меншої кількості тестів, незалежно від конкретних цілей процесу тестування, забезпечує глибину тестування, надає командам розробників та тестувальників можливостей застосування комбінаторних методів розробки тестів для створення структурованих варіацій у своїх тестових випадках [156-159].

Виділяються наступні підходи до тестування програмного забезпечення [160]:

1. «Статичне, динамічне та пасивне тестування». Безпосереднє виконання програмного коду із попередньо заданими тестовими випадками називається «динамічним тестуванням»[161].

Процес статичного тестування є «неявним», як і процес корегування. У випадках використання для перевірки структури висхідного коду інструментів програмування/текстових редакторів коду або перевірки синтаксису і потоку даних компіляторами (пре-компілятори), використовуючи методи та технології статичного аналізу програм. За допомогою прогамного засобу Sonar Lint (Додаток Д). Процес динамічного тестування використовується під час роботи самого застосунка. Існує можливість того, що процес динамічного тестування буде розпочато до 100% завершення роботи застосунка для організації тестування окремих розділів висхідного коду та до окремих модулів застосунка. Для цього використовуються «заглушки»/ «драйвери» або застосунок виконується безпосередньо із «середовища налагоджувача» (англ. debugger environment). Статичне тестування включає інструменти

«верифікації» застосунків, у той час, як у динамічному тестуванні, застосовуються засоби та механізми «валідації» [162].

Пасивне тестування перевіряє поведінку системи без організації будь-якої взаємодії з програмним застосунком. На відміну від активного тестування, задача тестувальників полягає не у наданні тестових даних, а у перегляді системних журналів та трасуванні. Тестувальники шукають шаблони та конкретну поведінку для прийняття відповідних рішеннь. Це пов’язано з перевіркою часу роботи в автономному режимі та аналізом журналів.

Даний підхід було застосовано для контролю якості модулів: «авторизації», «особистий кабінет» та «головна сторінка».

2. Дослідницький підхід. (англ., Exploratory approach). «Дослідницьке тестування — це підхід до тестування програмного забезпечення, який коротко описується як одночасне навчання, проєктування тесту та його виконання» [162]. Джем Канер, який ввів цей термін у 1984 році, визначив «дослідницьке тестування» як «стиль тестування програмного забезпечення, який наголошує на особистій свободі та відповідальності окремого тестувальника щодо постійної оптимізації якості його/її роботи за допомогою навчання, пов’язаного з тестуванням» [163]. Дизайн тесту, виконання тесту та інтерпретація результатів тесту розглядаються як взаємодоповнені дії, які виконуються паралельно протягом усього проекту. Мета дослідницького тестування — з’ясування того, як застосунок працюватиме в робочому середовищі, і пошук відповіді на запитання про те, як на роботу застосунка впливатимуть задачі різного рівня складності. Якість тестування залежить від кваліфікації тестувальника у винаходженні тестових випадків та виявленні дефектів. Чим більшим об’ємом інформації про застосунок і чим більшою кількістю різних методів тестування володіє тестувальник, тим ефективнішим буде процес тестування, та інформативність його результатів.

Під час проведення дослідницького тестування очікувані результати є відкритими для команд розробників та тестувальників [164]. Функціональні задачі тестувальника дослідницького тестування:

- виконання налаштування середовища тестування;
- робота з тестовим середовищем, тестовими даними;
- спостереження та оцінка застосунка та його поведінки;
- критичне дослідження результатів;
- повідомлення та запис інформації, яка є можливою помилкою (що загрожує цінності продукту для певної особи) або проблемою (яка загрожує якості тестування).

Тестування у сучасних процесах розробки є комбінацією дослідницького та сценарного тестування, в залежності від контексту. Основні переваги дослідницького тестування [163]:

- необхідність або нижчому рівні підготовки команди тестування;
- швидкість виявлення важливих помилок;
- на етапі виконання дослідницьке тестування, як правило, є більш інтелектуально заохочуваним, ніж виконання сценарних тестів;
- використання тестувальниками дедуктивних міркувань на основі результатів попередніх тестів для коригування напрямку майбутніх процесів тестування паралельно із процесом розробки застосунку;
- після початкового тестування більшість помилок виявляються шляхом певного дослідницького тестування. Це логічно демонструється наступним твердженням: «Програми, які проходять певні тести, як правило, продовжують проходити ті самі тести та з більшою ймовірністю не пройдуть інші тести чи сценарії, які ще належить дослідити» [165-168].

Недоліки дослідницького тестування:

- тести, створені та виконані на льоту, не можуть бути перевірені заздалегідь (тим самим, неможливим стає запобігання виникнення помилок у висхідному коді та тестових випадках);
- складність демонстрації того, які саме тести були запущені.

Цей підхід застосовано у процесі впровадження тестового середовища, docker контейнерів та сервісу журналу подій (Graylog).

3. Попереднє тестування проти адаптивного тестування. Тип стратегії тестування, який буде виконуватися, залежить від того, чи існує необхідність у визначенні тих тестів, які будуть застосовані до початку виконання плану тестування (попередньо встановлене тестування), чи кожен вхід, який буде застосовано, буде динамічно залежним від результатів, отриманих під час застосування попередніх тестів (адаптивне тестування).

Цей тип тестування було застосовано під час розробки back-end компонентів [169-171].

4. Підхід «скриньок». Традиційно методи тестування програмних застосунків розподіляються на тестування «білої скриньки» та тестування «чорної скриньки». Підхід «тестування скриньками» використовуються у випадках, коли необхідно враховувати точки зору, яких тестувальник буде дотримуватись під час розробки тестових випадків. Існує також і гібридний підхід — «тестування сірого ящика». Оскільки концепція тестування в сірому ящику, за якої тести розробляються відповідно до певних, конкретних елементів дизайну, набуває популярності, це «довільне розрізнення» між тестуванням у «чорній та білій скриньках» поступово зникає.

Задачі тестування «білої скриньки» (також відоме як тестування «прозорої скриньки», тестування «скляної скриньки» та «структурне тестування»):

- аналіз внутрішньої структури застосунка;

- пошук відмінностей у роботі застосунка від функціональних вимог, відкритих для кінцевого користувача пердставлено на рис. 2.2.

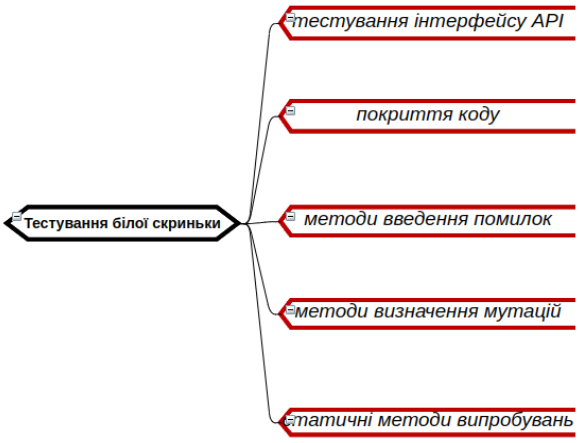


Рисунок 2.2. Методи, які використовуються в тестуванні «білої скриньки»

У тестуванні «білої скриньки» для розробки тестів використовуються як внутрішня перспектива системи (висхідний код), так і відповідні навички програмування. Вхідні дані, необхідні для проходження коду, обираються тестувальником, на їх основі визначаються відповідні висхідні дані. Це практично аналогічно тестуванню вузлів електронної схемі. Тестування «білої скриньки» застосовується як на рівнях модуля, інтеграції та системного рівня всього процесу тестування, так і на рівні усього пристрою [172]. Тестування перевіряє:

- шляхи всередині блоку під час упровадження та роботи;
- шляхи між блоками під час упровадження та роботи;
- шляхи між підсистемами під час тестування на системному рівні.

Незважаючи на те, що цей метод тестування застосовується для виявлення великої кількості помилок або проблем, він не спроможний виявити нереалізовані частини програмної специфікації або хибні, чи взагалі відсутні вимоги.

Інструменти покриття коду дають змогу оцінити повноту набору тестів, створених за допомогою будь-якого методу, включаючи тестування «чорної скриньки». Вони надають групі програмістів інструменти для проведення дослідження частин системи, які рідко тестуються, і гарантують, що найбільш важливі функціональні вимоги були перевірені. «Охоплення коду» у вигляді показника якості програмного забезпечення аналізується у відсотках для:

– покриття функцій, яке звітує про виконані функції;

– покриття оператора, яке звітує про те, скільки рядків, було необхідно виконати для завершення тесту;

– покриття рішення, яке повідомляє про те, виконання гілки True і False даного тесту;

– 100% покриття операторів гарантує, що всі шляхи виконання усього коду або однієї певної «гілки» (якщо береться до уваги саме поток керування) будуть виконані хоча б один раз. Відсоток покриття є корисним покажчиком для:

– забезпечення коректної роботи застосунка;

– впевненості у повному задоволенні функціональних вимог.

5. Тестування методом «чорної скриньки». У тестуванні методом «чорної скриньки», яке також часто називають «функціональним тестуванням», програмне забезпечення розглядається як «чорна скринька», його функціональність перевіряється без будь-якої інформації про внутрішню архітектуру застосунка, без перегляду вихідного коду [173]. Команді тестування доступна лише інформація про формати та діапазони можливих значень результатів роботи програмного застосунка, а не про архітектурні рішення та шляхи отримання застосунком цих результатів. Методи тестування «чорної скриньки» зазначені на рис.2.3.

При застосуванні «тестування на основі специфікацій» тестувальники перевіряють саме те, як програмне забезпечення функціонує відповідно до застосованих вимог. На цьому рівні тестування

вимагає ретельного плану тестування, що надається тестувальнику, який у процесі роботи перевіряє, чи дійсно для даного вхідного сигналу висхідне значення або поведінка відповідає очікуваному значенню, вказаному у тестовому випадку.

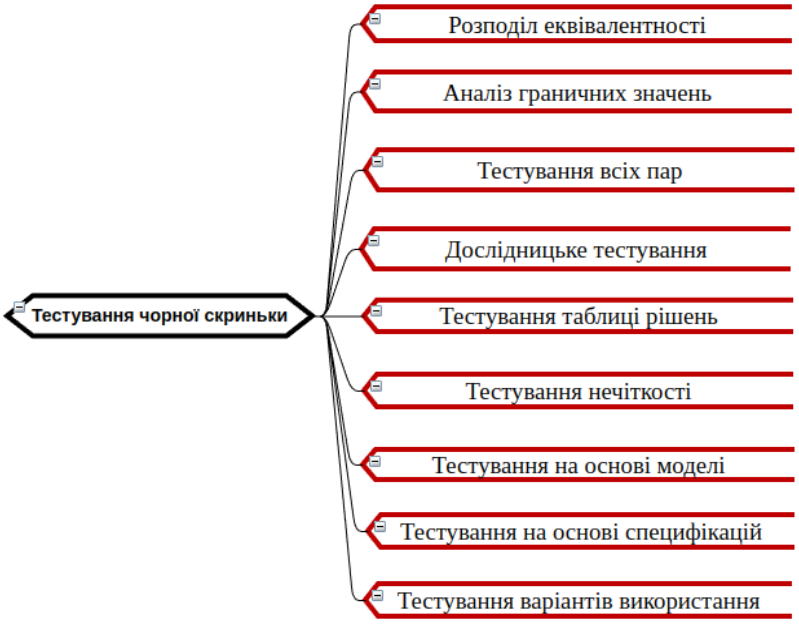


Рисунок 2.3. Методи тестування «чорної скриньки»

Тестові випадки будуються навколо специфікацій і функціональних вимог. Для цього тестувальником застосовуються «зовнішні описи» [174] програмного забезпечення які включають:

- специфікації;
- вимоги;
- проєкти для отримання «тестових випадків».

Отримані в результаті «зовнішніх описів» тести можуть бути як функціональними, так і нефункціональними.

Тестування на основі специфікацій використовується для забезпечення коректного функціонування системи, але є недостатнім для захисту від складних ситуацій або ситуацій, пов’язаних із високим ризиком.

Головна перевага техніки «чорної скриньки»: від тестувальника не вимагаються знання або навички з програмування.

Тестування на основі специфікацій застосовується на всіх рівнях тестування програмного забезпечення:

- модульному;
- інтеграційному;
- системному;
- прийнятному.

Разом із методом тестування на основі специфікацій застосовується більшість методів тестування на «вищих рівнях» [175].

6. Тестування інтерфейсу компонентів. Тестування інтерфейсу компонентів є різновидом тестування «чорної скриньки», яке зосереджене на значеннях даних, а не на відповідних діях обраного компонента певної підсистеми. Метод тестування інтерфейсів компонент використовується для перевірки та обробки інформації, яка передається між різними блоками або складовими частинами обраної підсистеми, проте без необхідності у повному тестуванні взаємодії між обраними складовими частинами. Інформація, яка передається, розглядається як «пакети повідомлень», а діапазони припустимих значень або відповідність значень необхідним типам даних перевіряються на присутніх даних, які згенеровані одним “пакетом”, після чого виконуються перевірки на повноту та дійсність до того як інформація буде передана до іншого блоку.

Наприклад, наступний варіант тестування інтерфейсу: зберігається окремий файл системного журналу даних, який містить записи про всю інформацію, що передається у системі із відповідною міткою часу, що дозволяє аналізувати тисячі випадків передачі даних між блоками впродовж дня, тижня, місяця або навіть року. До тестів включається перевірка обробки певних окремих «екстремальних значень даних», у той час як інші змінні елементи інтерфейсу передаються у вигляді «нормальних» значень.

7. «Візуальне тестування». Мета даного методу тестування — надання розробникам можливостей для перевірки станів застосунка в

момент виникнення збою програмного забезпечення, шляхом надання інформації у вигляді, який дозволяє розробнику застосувати ефективні алгоритми пошуку потрібної інформації. Візуальне тестування засновано на припущені того, що саме візуальна демонстрація спостерігачу знайденої проблеми або невдачі тесту, а не надання її текстового опису, значно підвищує ясність [176] і розуміння причини виникнення проблеми. Тому процес візуального тестування вимагає запису повністю всього процесу тестування – фіксації усіх дій тестувальника у тестовій системі у форматі відео. Вихідні відео доповнюються вхідними даними тестувальника в реальному часі через вебкамеру та аудіо-коментарями з мікрофонів.

Візуальне тестування надає ряд переваг:

- збільшення «якості зв'язку» між командами розробників та тестувальників, оскільки тестувальники наочно демонструють проблему і події, що призвели до неї, розробнику, замість надання звичайного опису проблеми
- усунення потреби відтворення помилок тестування на великому наборі тестових випадків;
- доступність для розробника усіх необхідних доказів невдачі тесту.

При необхідності перевірки цілісності програмного забезпечення, найбільш важливі методи — спеціальне тестування та дослідницьке тестування, оскільки ці методи потребують значно меншої кількості ресурсів на підготовку до реалізації, у той час як процес виявлення критичних помилок значно пришвидшується.

8. Тестування методом «сірої скриньки». Тестування методом «сірої скриньки» ящика передбачає зі сторони команди тестування, володіння інформацією про внутрішні структури даних та алгоритми для цілей розробки тестів. Тестувальник має доступ як до «висхідного коду застосунка», так і до скомпільованого «двійкового файлу», який

виконується у тестовому середовищі. Тестування методом «сірої скриньки» також використовує засоби «зворотнього проєктування» (англ. «Reverce Engineering») разом із застосуванням засобів, які дозволяють проведення динамічного аналізу висхідного коду для визначення граничних значень. Маніпуляції із вхідною інформацією та форматування висхідної інформації не кваліфікуються як «сіра скринька», тому що вхідна та висхідна інформація існує «поза межами» «чорної скриньки», яка в літературі має назву «тестована система». Подібна відмінність особливо важлива під час проведення "інтеграційного тестування" між двома модулями висхідного коду, реалізованих двома або більшою кількістю різних розробників, а також у випадку, в якому для тестування доступні лише інтерфейси модулей [175-176].

Базуючись на концепціях роботи програмного забезпечення, тестувальник робить обґрунтований вибір необхідної на поточному етапі життєвого циклу застосунка методології тестування під час «зовнішнього» тестування. Тестувальнику, що працює методом «сірої скриньки», дозволяється налаштування власного, повністю ізольованого від інших користувачів тестового середовища з такими передумовами, як автоматичне заповнення тестової бази даних. Тестувальник має можливість вести спостереження стану продукту, який тестується, після виконання наступних дій у системі:

- виконання окремих SQL- операторів у базі даних;
- виконання SQL-запитів у базі даних, для переконання у відображені необхідних очікуваних змін. Тестування в сірому ящику реалізує інтелектуальні сценарії тестування на основі обмеженої кількості інформації. Особлива увага приділяється до обробки типів даних та обробки виняткових ситуацій тощо.

В літературі виділяється щонайменше три рівні тестування, зазначені на (рис 2.4.)[175].

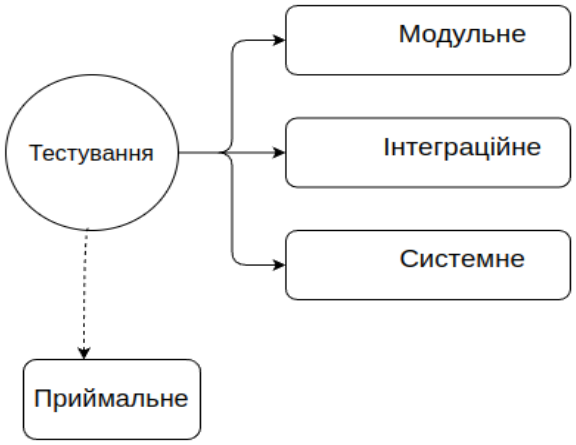


Рисунок 2.4. Рівні тестування

Також деяких статтях [172-173] додається четвертий рівень: приймальне тестування. Виділяються три форми приймального тестування:

- форма робочого приймального тестування;
- форма тестування кінцевим користувачем (бета-версії);
- форма тестування, зосереджена на перевірці відповідності застосунка функціональним очікуванням.

Тести згруповуються до одного із наведених рівнів за місцем додавання в процес розробки програмного забезпечення або за специфічністю додаваємого теста.

9. Модульне тестування у комп'ютерних науках — це метод тестування програмного забезпечення, за допомогою якого окремі одиниці висхідного коду (набори одного чи кількох модулів комп'ютерного застосунка) разом із пов'язаними керуючими даними, процедурами використання та робочими процедурами перевіряються для визначення, придатності для застосування. Це стандартний крок у сучасних підходах до розробки та впровадження програмних засобів, таких як Agile [177].

Модульне тестування — технологія тестування, яка перевіряє функціональність виділеної частини коду на функціональному рівні. В об'єктно-орієнтованому середовищі тестування відбувається на рівні

класів, а до складу мінімальних модульних тестів входять методи «конструктор» та «деструктор».

Модульні тести створюються розробниками під час роботи над кодом (у стилі «білої скриньки»), для забезпечення відповідності вхідних та вихідних параметрів функцій системи функціональним вимогам. Для однієї функції утворюється необхідна кількість тестів, призначенням яких є виявлення кутових випадків та розгалужень у коді. Одиничне тестування не здатне повністю перевірити функціональність частини програмного засобу. Тестування використовується для перевірки коректності роботи окремих складових частин застосунка незалежно один від одного.

«Модульне тестування» входить до складу процесу розробки програмного забезпечення, який, у свою чергу, передбачає одночасне та злагоджене використання широкого набору різноманітних стратегій спрямованих на:

- запобігання, виявлення та знешкодження дефектів;
- зменшення ризиків, часу і витрат на розробку;
- впровадження програмного застосунка в цілому.

«Модульне тестування» організується, підтримується та виконується розробником або інженером програмного забезпечення на перших етапах будівництва «життєвого циклу розробки програмного забезпечення» [178]. Мета модульного тестування — усунення помилок та недоліків конструкції застосунка до того, як висхідний код буде впроваджено, або переведено на додаткові етапи тестування.

Стратегія модульного тестування спрямована на підвищення якості кінцевого програмного засобу, та підвищення ефективності процесу розробки загалом.

У залежності від потреб та очікувань організації щодо розробки програмного засобу поряд із модульним тестуванням використовуються наступні методи:

- методи та засоби статичного аналізу коду;

- методи аналізу потоків даних;
- методи аналізу метрик;
- методи рецензування коду;
- методи аналізу покриття коду.

Мета модульного тестування — відокремлення частин застосунка та доведення відповідності результатів роботи цих відокремлених частин функціональним вимогам. Модульне тестування забезпечує суворий письмовий договір, якому повинен відповідати фрагмент коду. Переваги модульного тестування [179] зазначено в таблиці 2.7.

Таблиця 2.7.

Переваги модульного тестування

Раннє виявлення проблем циклу розробки	Модульне тестування виявляє проблеми на ранніх стадіях циклу розробки. Проблеми включають як помилки в реалізації припущені програмістами, так і недоліки або відсутні частини специфікації застосунка. Процес написання ретельного набору тестів змушує автора продумувати входи, виходи та умови помилок і, таким чином, більш чітко визначати бажану поведінку застосунка [178].
Знижена вартість	Вартість виявлення помилки до початку кодування або під час першого етапу життєвого циклу застосунка значно нижча, ніж вартість виявлення, ідентифікації та виправлення помилки на наступних етапах життєвого циклу. Помилки у випущеному програмному продукті спричиняють коштовні проблеми для кінцевих користувачів програмного забезпечення. Код може бути непридатним або надскладним для модульного тестування, якщо він написаний не відповідно до міжнародних стандартів написання програмного коду,

	тому модульне тестування додатково «примушує» розробників краще структурувати функції та об’єкти.
Розробка, що керується тестуванням	У підході «розробка, що керується тестуванням» (англ. Test Driven Development, скор. TDD), який використовується в методології scrum, першим етапом розробки програмного забезпечення стає розробка модульних тестів. Тільки після успішного виконання модульних тестів, висхідний код вважається «завершеним», тобто готовим до передачі на подальші етапи циклу розробки [177]. Модульні тести використовуються, оскільки більша база коду розробляється або під час зміни коду, або через автоматизований процес зі збіркою. У разі неуспішного виконання модульних тестів, висхідний код вважається таким, що містить помилку. Тести дозволяють відстеження конкретного місця несправності. Задачі модульного тестування - попередження команди розробників про виникнення проблеми до передачі висхідного коду команді тестування або клієнтам.
Прискорений випуск нових версій застосунка	Модульне тестування дозволяє прискорити випуск нових версій при розробці програмного забезпечення. Тестуючи окремі компоненти, розробники максимально швидко виявляють проблемні ділянки, що призводить до ефективної ітерації циклів випуску.
Дозволяє рефакторинг коду	Для розробника модульне тестування дозволяє : - проведення ефективного рефакторингу висхідного коду; - оновлення системних бібліотек;

	- переконання у коректності роботи відповідного модуля (наприклад, під час регресійного тестування). Процедура полягає в написанні тестових випадків для всіх функцій і методів для ідентифікації помилки, одразу після її виникнення [179].
Відстеження порушень	Модульні тести дозволяють виявлення змін, які періодично призводять до порушень проєктного контракту.
Зменшення невизначеності	Модульне тестування зменшує невизначеність у блоках програмного коду і використовується в стилі тестування «знизу-вгору». Якщо процес тестування роботи окремої частини застосунка передує процесу тестування роботи сукупності кількох частин, інтеграційне тестування стає набагато ефективнішим.
Документація поведінки системи	Модульне тестування надає «живу документацію» системи. Розробники, яким необхідно дізнатися про механізми роботи конкретної функції застосунка, звертаються до модульних тестів для отримання базової інформації відносно програмного інтерфейсу.

Модульні тести реалізують функціональні та технічні вимоги, які характеризують успішне завершення відповідного етапу життєвого циклу застосунка. Функціональні вимоги включають результати належного/неналежного використання відповідного модуля, а також можливу негативну поведінку модуля при некоректних вхідних параметрах [179]. На етапі впровадження модульного тестування автоматично формується документація критичних характеристик. При застосуванні методології “розробка керована тестуванням”, програмне забезпечення розробляється із використанням комбінації процесів

створення та впровадження модульних тестів для визначення програмного інтерфейсу функцій та процесів рефакторингу, необхідних після проходження відповідних тестів. Кожен модульний тест розглядається як окремий елемент архітектури застосунка, що визначає класи, методи та спостережувану поведінку.

Недолік модульного тестування — неспроможність до виявлення усіх функціональних та логічних помилок в програмі через відсутність інструментів для оцінки кожного можливого шляху виконання застосунка, крім найтривіальніших. Ця проблема відноситься до надмножини «проблеми затримки» (англ. halting problem), яка є нерозв'язаною. Те саме стосується модульного тестування. Модульне тестування за визначенням обмежене функціональністю відповідних модулів, через це, модульне тестування неспроможне до виявлення помилок інтеграції або помилок системного рівня (наприклад, функцій, які виконуються паралельно на декількох пристроях або нефункціональних тестових областей модулів, таких як продуктивність). Модульне тестування проводиться в поєднанні з іншими підходами тестування програмного забезпечення, оскільки призначення модульних тестів — показ наявності або відсутності окремих визначених у функціональних вимогах помилок. Задача доведення повної відсутності помилок у застосунку за допомогою модульних тестів є також нерозав'язною. Для забезпечення коректної поведінки для кожного шляху виконання та кожного можливого набору вхідних даних, а також забезпечення відсутності помилок, використовуються інші методи та підходи, а саме формальні методи верифікації, призначення яких – доведення відсутності неочікуваної поведінки в програмному компоненті [177-178].

Складнощі, пов'язані з проведенням модульного тестування, наведено в таблиці 2.8.

Таблиця 2.8.

Складнощі проведення модульного тестування в сучасних проєктах

Продумана ієрархія модульних тестів не дорівнює інтеграційному тестуванню	Інтеграція з периферійними пристроями — це задача інтеграційного тестування, яка не має відношення до модульних тестів. Інтеграційне тестування залежить від результатів ручного тестування. Повне тестування програмного забезпечення є комбінаторною проблемою. Наприклад, кожне логічне рішення вимагає щонайменше двох перевірок: одна з результатом «true» й інша з результатом «false». Як наслідок, кожний рядок програмного коду, вимагає від 3 до 5 рядків тестового коду. Розробка тестів у цьому випадку потребує часових витрат і призводить до збільшення кінцевої вартості застосунка. Модельне тестування має технічні обмеження, наприклад: неможливість перевірки програмного коду, що містить недетермінований вибір або той, що виконується кількома потоками [173]. Крім того, сам код модульного тесту може містити логічні або технічні помилки.
Складнощі із налаштуванням реалістичних і корисних тестових випадків	Головна проблема, пов’язана із написанням модульних тестів — складність налаштування реалістичних і корисних тестових сценаріїв. Створені тести мають відповідати початковим умовам для забезпечення очікуваної поведінки застосунка, що тестується, або його окремого модуля. Якщо ці початкові умови є не коректними, тест не виконуватиме код у реалістичному

	контексті, що зменшує цінність і точність результатів модульного тесту.
Вимагає дисципліни протягом усього процесу розвитку програмного продукту	Для отримання запланованих переваг від модульного тестування необхідна сувора дисципліна серед членів команди розробників протягом усього процесу розробки програмного забезпечення.
Вимагає контроль версій	Важливо ведення ретельного обліку не лише проведених тестів, а й усіх змін, внесених у вихідний код поточної та будь-якої іншої функціональної частини програмного забезпечення, що вимагає використання системи контролю версій. У випадку непроходження конкретного тесту пізнішою версією модуля, який був успішно пройденим попередніми версіями, програмне забезпечення для контролю версій надає список змін у вихідному коді (за їх наявності), за обраний проміжок часу [177].
Потребує регулярних оглядів	Запровадження стійкого процесу забезпечення регулярного перегляду та вирішення невдалих тестових випадків.
Обмеження для вбудованого системного програмного забезпечення	Модульне тестування вбудованого системного програмного забезпечення представляє унікальну проблему; оскільки вбудовані системні застосунки були розроблені на платформі, відмінній від тієї, на якій вони зрештою працюють [178].

Обмеження для тестування інтеграції із зовнішніми системами	Найпростіші модульні тести мають наступний вигляд: порівняння вхідних параметрів із заданим еталоном та порівняння висхідних параметрів із заданим еталоном. Створення модульних тестів, які перевіряють функції методу, який взаємодіє із зовнішнім середовищем по відношенню до програми, вимагає додаткових апаратних та програмних ресурсів. Наприклад, для перевірки методу, який працюватиме з базою даних, необхідно створення макета взаємодії із копією відповідної бази даних.

10. Інтеграційне тестування (інша назва — інтеграція та тестування, скорочено I&T) — це етап тестування програмного забезпечення, на якому повністю тестується програмний модуль або у випадку, якщо він складається з кількох програмних модулів, ці відповідні модулі об’єднуються, після чого тестуються як група. Задача інтеграційного тестування — проведення оцінки відповідності системи або компонента визначеним функціональним вимогам. Етап інтеграційного тестування відбувається після етапу модульного тестування, перед вирішальним етапом кінцевого тестування системи в цілому. Вхідні параметри інтеграційного тестування — модулі системи, які пройшли етап модульного тестування. Ці параметри ґрунтуються за функціональністю та застосовують тести, визначені в плані інтеграційного тестування для цих відповідних груп, задача яких надати інтегровану систему, готову до системного тестування [172]. Основні типи інтеграційного тестування наведені на рис. 2.5.

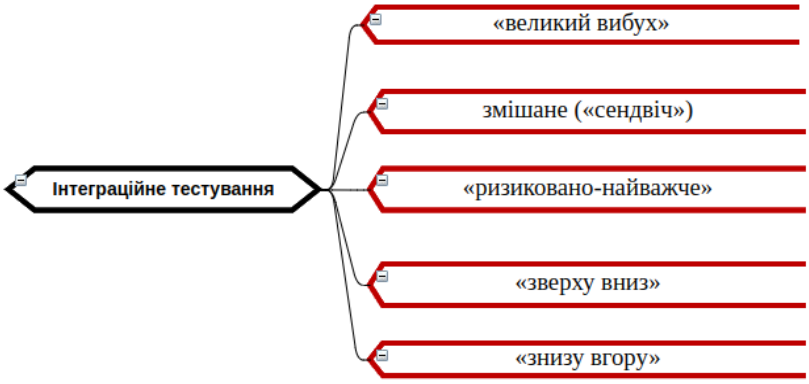


Рисунок 2.5. Типи інтеграційного тестування

Тестування «*великого вибуху*» передбачає об'єднання більшості розроблених модулів, для формування повної програмної системи або основної частини системи для подальшого інтеграційного тестування. Цей метод є ефективним з точки зору економії часу в процесі інтеграційного тестування. Неналежне документування тестових випадків та відповідних результатів призводить до ускладнення всього процесу інтеграції, що потенційно перешкоджає команді тестування у досягненні мети інтеграційного тестування [173].

Тестування «*знизу-вгору*» передбачає перевірку, у першу чергу, компонент найнижчого рівня із подальшим використанням отриманих результатів для полегшення тестування компонент вищого рівня. Процес повторюється доки не буде завершено перевірку компонент на вершині ієрархії. Усі нижні або низько рівневі модулі, процедури чи функції інтегруються перед тестуванням. Після завершення інтеграційного тестування відповідних модулів нижчого рівня формується наступний рівень модулів, який призначено для використання на етапі інтеграційного тестування. Цей підхід є корисним у випадку, коли розробка всіх або більшості модулів одного рівня є завершеною. Інша задача цього метода – визначення рівня завершеності розробленого програмного забезпечення та полегшенні звітування про хід тестування у вигляді відсотків.

Під час тестування «*зверху-вниз»* першими перевіряються верхні інтегровані модулі, а потім відповідні гілки тестованого модуля крок за кроком.

«Сендвіч-тестування» поєднує тестування «зверху-вниз» і тестування «знизу-вгору». Обмеження сендвіч-тестування — неможливість перевірки умов, не зазначених в інтеграційних тестах.

11. Системне тестування — це тестування повної інтегрованої системи, яке проводиться для оцінки відповідності системи встановленим вимогам.

Вхідні дані системного тестування – всі інтегровані компоненти, які пройшли етап інтеграційного тестування. Мета інтеграційного тестування – виявлення будь-яких невідповідностей між об’єднаними одиницями, так званими «збірками». Системне тестування, спрямоване на виявлення дефектів як усередині «збірок», так і в системі в цілому. Результат — поведінка, реалізована або спостережувана під час тестування окремого компонента чи системи загалом. Системне тестування виконується на всій системі в контексті специфікацій функціональних вимог (англ. Functional Requirements Spesification (FRS)) або специфікацій системних вимог (англ. System requirements Spesification SRS), або об’єднанні обох специфікацій. Системне тестування перевіряє архітектуру, поведінку і вірогідність очікувань клієнтів [173]. Системне тестування призначене для перевірки границь, визначених у специфікаціях вимог до програмного або апаратного забезпечення. Три підходи до системного тестування наведені на (рис 2.6.):

– *руйнівне тестування*: тестування виконується ітераційно до тих пір поки система не припинить роботу для оцінки стабільності застосунка або поведінки системи під різними навантаженнями;

– *неруйнівне тестування*: методи аналізу для оцінки властивостей компонентів або системи без викликання ситуації припинення роботи системи;

– *ін'єкція помилки*: техніка тестування, яка навантажує систему незвичним способом для перевірки поведінки системи.

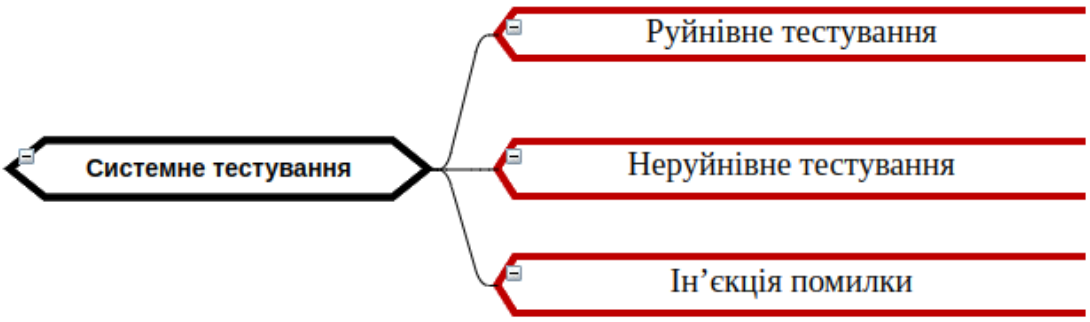


Рисунок 2.6. Підходи до системного тестування

Методи системного тестування розділяються за наступними предметними областями:

12. Тестування програмного забезпечення розглядається як дослідження, яке проводиться зацікавленими сторонами для отримання інформації відносно кінцевої якості програмного продукту або послуги, що тестується. Мета тестування програмного забезпечення – надання об'єктивного, незалежного погляду на програмне забезпечення, для забезпечення можливостей проведення оцінок та отримання висновки відносно можливих ризиків впровадження програмного застосунка для компанії. Тестування програмного забезпечення передбачає виконання компонент застосунка або системної компоненти для проведення оцінки однієї або кількох необхідних властивостей. Сукупність цих властивостей вказує на рівень відповідності компонент системи функціональним вимогам, які використовувалися під час проєктування і розробки системи. Через те, що кількість можливих тестів, навіть, для найпростіших компонент програмного застосунка є практично нескінченною, усі підходи до тестування програмного забезпечення використовують певну стратегію для обрання конкретних тестів, які будуть виконуватись за наявний час і ресурси.

13. ***Тестування мобільних пристроїв****.* Тестування мобільних пристроїв перевіряє якість роботи застосунка на мобільних пристроях, таких як мобільні телефони, КПК тощо. Тестування проводиться на апаратному і на програмному рівнях. Тестування мобільних пристроїв включає в себе тестування R&D, заводське тестування та тестування сертифіката [174]. Тестування мобільних пристроїв включає низку дій від моніторингу та усунення несправностей мобільних додатків, вмісту та послуг на реальних телефонах.

У теорії інженерії програмного забезпечення **приймальне тестування** - це випробування, яке проводиться для визначення відповідності застосунка технічним вимогам специфікації або контракту. У системній інженерії програмного забезпечення це включає тестування системи перед її впровадженням. У теорії тестування програмного забезпечення приймальне тестування визначається наступним чином: формальне тестування застосунка щодо потреб користувачів, вимог і бізнес-процесів, що проводиться для визначення рівня задоволення критеріям прийнятності системи і надання користувачам, клієнтам або іншим уповноваженим організаціям визначення прийняття системи.

Останній етап життєвого циклу програмного забезпечення – тестування прийнятності користувачем, проводиться безпосередньо перед остаточним випуском для оцінки можливостей програмного продукта працювати з реальними сценаріями. Відтворюючи поведінку користувача, тестове середовище перевіряє відповідність системи до бізнес-вимог і відхиляє зміни у випадку не виконання певних конкретних критеріїв.

Критерії прийнятності – це критерії, яким система або компонент повинні задовольняти, для прийняття користувачем, клієнтом або іншою уповноваженою організацією.

У випадках, якщо всі тестові випадки не були виконані протягом однієї, ітерації, тест, набір приймальних тестів необхідно виконати кілька разів.

Рисунок 2.7. Форми приймального тестування

Набір приймальних тестів виконується із використанням попередньо визначених процедур приймальних тестів для надання вказівок тестувальникам, які дані використовувати, покрокові процеси, яких слід дотримуватися, і очікуваний результат після виконання [173]. Фактичні результати зберігаються для порівняння з очікуваними результатами. У випадку збігу фактичних результатів з очікуваними результатами для кожного тестового прикладу, тестовий приклад вважається успішним. Якщо кількість неуспішних тестових випадків не перевищує попередньо визначеного у проєкті порога, набір тестів вважається успішним. У цьому випадку система відхиляється або приймається на умовах, попередньо погоджених замовником і виробником. Очікувані результати успішного виконання тесту:

– тестові випадки виконуються для заздалегідь визначених даних;

– фактичні результати фіксуються;

– фактичні та очікувані результати порівнюються і визначаються результати тестування.

Мета приймального тестування – забезпечення впевненості, що розроблений продукт відповідає як функціональним, так і нефункціональним вимогам [174]. Мета проведення приймального тестування після його завершення та за умови дотримання критеріїв прийнятності замовника, як очікується, визначення розробки/покращення продукту таким, що відповідає визначеним вимогам, попередньо узгодженим між замовником та постачальником/розробником продукту .

Приймальне тестування користувачем – процес перевірки відповідності застосунка функціональним вимогам користувача. Приймальне тестування проводиться призначеним кінцевим користувачем або експертом у відповідній галузі, який після тестування надає скорочений виклад результатів для підтвердження. В інженерії програмного забезпечення приймальне тестування користувачем – один із завершальних етапів життєвого циклу проєкта, який відбувається до моменту, прийняття клієнтом нової системи. Користувачі системи виконують тести відповідно до того, що відбувається у реальних сценаріях.

Важливою складової є надання тестувальнику матеріалів, максимально схожих на матеріали, які матиме кінцевий користувач. Тестувальники отримують сценарії з реального життя, наприклад, три найпоширеніших або складних завдань, які виконуватимуться користувачами.

Приймальне тестування користувачем – остаточна перевірка необхідної бізнес-функціональності та належного функціонування системи, шляхом імітації реальних умов від імені клієнта-платника або конкретного великого клієнта.

Користувальницькі тести, які виконуються клієнтами або кінцевими користувачами, не зосереджуються на виявленні простих «косметичних» проблем, (наприклад, орфографічних помилок) та фатальних збоях програмного забезпечення. Команди тестування та розробки виявляють і

виправляють ці проблеми під час попередніх етапів модульного тестування, інтеграційного тестування та тестування системи.

Приймальне тестування користувачем виконується за відповідними тестовими сценаріями. Тестові сценарії відрізняються від системних або функціональних тестів тим, що вони зосереджені не на перевірці окреми випадків, а на перевірці сукупності дій користувача, яка призводить до відповідного результату. Тестові сценарії гарантують, зосередження фокуса саме на послідовності дій, а не на технічних чи інших специфічних деталях системи, утримуючись від елементарних тестових кроків, для врахування відмінностей поведінок користувачів. Тестові сценарії розбиваються на логічні «дії», у яких змінюється «актор» (гравець/клієнт/оператор) або система (офіс, програмний інтерфейс).

За відповідними позначками та способами групування тестових підходів виділяються наступні типи тестування програмного забезпечення:

1. Тестування встановлення. До складу більшості систем програмного забезпечення входять процедури встановлення та налаштування, які необхідні перед використанням за основним призначенням. Тестування встановлення — перевірка цих процедур для досягнення зазначеної функціональності системи програмного забезпечення, яка може бути використана на робочому місті кінцевого користувача. До складу цієї процедури входить повне або часткове оновлення та процеси встановлення/видалення. Тестування встановлення шукає помилки, які виникають у процесі встановлення та впливають на прийняття користувачем і відповідає за перевірку здатності використання встановленого програмного забезпечення [173]. Є багато подій, які впливають на встановлення програмного забезпечення. Тестування встановлення перевіряє правильність встановлення, одночасно перевіряючи низку пов’язаних дій і подій. Наприклад:

– користувачеві необхідно вибрати кілька варіантів;

– залежні файли та бібліотеки необхідно виділити, завантажити або знайти;

– має бути присутня дійсна конфігурація обладнання;

– програмнмі системи можуть вимагати підключення до інших програмних систем.

Тестування встановлення також розглядається у якості підходу, заснованого на діях. Наприклад:

- встановлення програмного забезпечення різними способами, на різних типах систем;

- перевірка того, які файли додано або змінено на диску;

- перевірка коректності роботи встановленого програмного забезпечення.

Тестування встановлення виконується під час операційного приймального тестування інженером із тестування програмного забезпечення разом із менеджером конфігурації.

Тестування встановлення визначається як тестування, яке розміщує скомпільовану версію коду в середовищі тестування або передвиробничому середовищі, з якого він може або не може перейти до виробництва [172].

Нечітке посилання на тестування впровадження відбувається за межами середовища розробки програмного забезпечення, для обмеження пошкодження коду з інших майбутніх чи минулих версій застосунка або шляхом використання неправильної версії залежностей, таких як спільні бібліотеки, які знаходяться в середовищі розробки. Найпростішим підходом до тестування встановлення є запуск програми встановлення (інша назва — програмний пакет). Цей пакет програмного забезпечення використовує програму встановлення, яка діє як обгортка кількох конфігурацій і дозволяє встановити програмне забезпечення на різних апаратних середовищах і/або операційних середовищах. Кожна можлива конфігурація проходить належний рівень тестування.

У розподілених системах, зокрема у випадках випуску програмного забезпечення у вже діюче цільове середовище (наприклад, робочий вебсайт), інсталяція або розгортання програмного забезпечення передбачає зміни схеми бази даних, а також встановлення нового програмного забезпечення.

До планів розгортання за таких обставин включаються процедури відключення, використання яких призначене для повернення до попередньої версії цільового середовища, у випадку неуспішного розгортання [171]. План розгортання також тестується у середовищі, яке є копією реального середовища. Фактором, який підвищує організаційні вимоги до тестування встановлення, є необхідність синхронізації даних у тестовому середовищі розгортання з даними в реальному середовищі з мінімальними перешкодами. До цього типу впровадження включаються тестування процесів, що відбуваються під час встановлення або оновлення багаторівневої програми.

2. «Тестування на сумісність». Одна з найпоширеніших причин виникнення збою програмного забезпечення (справжнього чи уявного) – несумісність із вже встановленим на комп’ютер програмним забезпеченням, з операційними системами або окремими версіями цих систем, старими чи новими, або середовищами виконання, які суттєво відрізняються від оригінальних (наприклад: консольний термінал або застосунок із графічним інтерфейсом користувача, який призначено для запуску з робочого стола, перетворюється на вебзастосунок, який відображається у веббраузері). Наприклад, у випадку відсутності зворотної сумісності, однією з причин виникнення “помилки сумісності” є розробка і тестування програмного забезпечення лише в останній версії цільового середовища. Це призводить до ненавмисних наслідків: остання версія програмної системи не працює, або працює не коректно із попередніми версіями цільових середовищ, або на застарілому обладнанні, яке могло використовуватися у попередніх версіях цільових

середовищ. Періодично подібні проблеми вирішуються шляхом проактивного абстрагування функціональності операційних систем в якості окремого програмного модуля або бібліотеки. Методологія «тестування на сумісність» є частиною нефункціонального тестування прикладного програмного забезпечення, що забезпечує перевірку на сумісність застосунка із різними обчислювальними середовищами.

У комп'ютерній науці сумісність визначається як характеристика або ступінь, до якого програмна система здійснює обмін інформацією з іншими системами, використовуючи однакове програмне та апаратне забезпечення [173].

Співіснування – ступінь, до якої продукт ефективно виконує необхідні функції, користуючись спільним середовищем і ресурсами з іншими продуктами, без шкідливого впливу на будь-який інший продукт, а сумісність — це ступінь, до якого дві або більше систем, продуктів або компонент обмінюються інформацією та використовують інформацію, якою обмінювалися. У цьому контексті тестування на сумісність — збір інформації про продукт або програмну систему для визначення ступеня співіснування та сумісності, що демонструється в тестованій системі.

3. Тестування методом «Дим і розсудливість». У комп’ютерній галузі та теорії тестування програмного забезпечення димове тестування (інші назви: тестування довіри, тестування працездатності, тест верифікації збірки і тест приймання збірки) – попереднє тестування, або тестування працездатності, спрямоване на виявлення елементарних помилок, які мають достатньо серйозні наслідки для відхилення майбутніх випусків програмного забезпечення [167].

Димові тести – це підмножина тестів, яка охоплює найважливіші функції компонента чи системи та використовується для оцінки коректності роботи основних функцій програмного забезпечення.

Димові тести використовуються для визначення необхідності

проведення для застосунка подальшого, більш детального тестування (інша назва димового теста – попередній тест, або впускний тест).

Димові тести – це набір тестів, які виконуються на кожній новій збірці продукту для переконання у придатності збірки для тестування, до передачі збірки команді тестувальників. У парадигмі DevOps використання кроку тестування верифікації збірки — одна із ознак безперервної стадії зрілості інтеграції.

Наприклад, димовий тест надає відповідь на такі основні запитання:

- «чи працює програма?»;
- «чи відкривається інтерфейс користувача?»;
- «чи натискання головної кнопки щось робить?».

Мета процесу димового тестування – визначення рівня придатності застосунка для подальшого тестування. Як сказано в посібнику «Lessons Learned in Software Testing» [168]: «димові тести широко охоплюють функції продукту за обмежений час, якщо ключові функції не працюють або якщо ключові помилки ще не виправлено, ваша команда не буде витрачати час на встановлення чи тестування».

Час виконання димових тестів є відносно невеликим, що дозволяє надання швидшого зворотного зв’язку. Часта повторна інтеграція з димовим тестуванням є однією з найкращих галузевих практик. Кожна фіксація репозиторію висхідного коду запускає збірку безперервної інтеграції для забезпечення якомога швидшого виявлення регресії.

Якщо процес збірки займає забагато часу, існує можливість об’єднання кількох попередніх збірок в одну.

Димове тестування також проводиться тестувальниками перед прийняттям збірки для подальшого тестування. Microsoft стверджує, що після перевірки коду «димове тестування є найбільш економічно ефективним методом виявлення та усунення дефектів програмного забезпечення».

Димові тести можуть бути функціональними або модульними. Функціональні тести виконують повну програму з різними вхідними даними, включають сценарну серію вхідних даних програми можливо, навіть з автоматизованим механізмом для керування рухами миші.

Модульні тести виконують окремі функції, підпрограми або об'єктні методи та можуть бути реалізовані або у вигляді окремих функцій у самому коді, або у вигляді рівеню драйвера, який посилається на код, не змінюючи його.

4. Регресійне тестування – це повторне виконання функціональних і нефункціональних тестів для переконання у належній роботі протестованого програмного забезпечення після внесення змін. Негативний результат цього процесу називається регресією. Зміни, які потребують регресійного тестування:

- виправлення помилок;
- удосконалення програмного забезпечення;
- зміни конфігурації;
- заміну електронних компонентів апаратного забезпечення.

Оскільки набори регресійних тестів зростають із кожним виявленим дефектом, вони використовуються в процесах автоматизації тестування [169-170]. Виняток – регресійне тестування графічного інтерфейсу користувача, яке виконується вручну.

Методи регресійного тестування:

- **«Повторно перевірити все».** Цей метод перевіряє всі тестові приклади поточного застосунка для перевірки його цілісності. Незважаючи на вартість даного підходу, оскільки він вимагає повторного запуска всіх випадків, він забезпечує відсутність помилок через змінений код.
- **Вибір регресійного тесту.** На відміну від методу «повторної перевірки всього», цей метод запускає частину набору тестів через

вартість повторного тестування усієї сукупності, якщо вартість вибору частини тестового набору менша, ніж техніка «повторно перевірити все».

- **Пріоритизація тестових випадків.** Тестові випадки розташовуються за пріорітетами для підвищення швидкості виявлення помилок у сукупності тестів. Методи пріоритизації тестових випадків зосереджуються на плануванні тестових випадків таким чином, що тестові випадки з вищим пріоритетом виконуються перед тестовими випадками з нижчим пріоритетом.

Типи пріоритизації тестових випадків:

- *загальна пріоритизація* визначає пріоритетність тестів, які будуть корисними в наступних версіях;
- *пріоритизація залежно від версії* визначає пріоритет тестових прикладів щодо певної версії програмного забезпечення.

Регресійне тестування використовується при внесенні змін у функціональність програмного забезпечення або при виявленні помилок у програмному забезпеченні [170].

Регресійне тестування досягається за допомогою кількох підходів, якщо дотримується підхід тестування всіх – це забезпечує впевненість, що зміни, внесенні до програмного забезпечення, не вплинули на існуючі функції, які не змінюються.

У гнучкій розробці програмного забезпечення, яка зумовлює відносно короткий життєвий цикл розробки програмного забезпечення, ресурси обмежені, а зміни в програмному забезпеченні дуже часті, регресійне тестування призводить до створення великої кількості непотрібних накладних витрат [171].

У середовищі розробки програмного забезпечення, яке має тенденцію до використання компонент «чорної скриньки» від третьої сторони, виконання регресійного тестування ускладняється, оскільки будь-яка зміна в компоненті третьої сторони заважатиме решті системи.

Регресійне тестування використовується не тільки для перевірки правильності застосунка, а також для відстеження якості результатів роботи застосунка. Наприклад, при розробці компілятора регресійне тестування здатне відстежити розмір коду та час, потрібний для компіляції та виконання набору тестів.

Регресійні тести класифікуються як функціональні або модульні. Функціональні тести відповідають за виконання повного застосунка для різних вхідних даних. Модульні відповідають за виконання окремих функцій, підпрограм або об'єктних методів. Й інструменти функціонального тестування, й інструменти модульного тестування р – автоматизовані й часто є продуктами сторонніх розробників, які не є частиною набору компіляторів.

Функціональний тест – запланована серія вхідних даних застосунка, можливо навіть, із залученням автоматичного механізму для керування рухами миші та натисканням.

Модульний тест – набір окремих функцій у самому коді або рівень драйвера, який пов'язується з кодом, не змінюючи його.

5. Альфа-тестування – це симуляція або фактичне операційне тестування потенційними користувачами/клієнтами чи незалежною командою тестувальників на відповідній вебсторінці сайту проєктної команди [149]. Метод альфа-тестування застосовується до вже готового програмного продукту у якості певної форми «внутрішнього приймального тестування» до того, як програмне забезпечення переводиться на наступний етап: бета-тестування.

6. Бета-тестування відбувається після етапу альфа-тестування і розглядається у якості однієї із форм зовнішнього тестування «прийнятності користувачем».

Відповідна версія програмного застосунка, що має назву «бета-версія», випускається для невеликої обмеженої кількості користувачів за межами команди програмістів та команди тестувальників.

Таких користувачів називають бета-тестувальниками. Ці групи користувачів отримують програмне забезпечення для того, щоб за рахунок подальшого тестування впевнетись у невеликій кількості недоліків або помилок у кінцевому продукті [165].

Подібні «Бета-версії» розповсюджуються серед відкритої громадськості для збільшення кола відгуків від максимально великої кількості майбутніх потенційних користувачів. Доступ до «Бета-версії» залишається відкритим протягом тривалого або, навіть, невизначеного періоду часу («постійна бета-версія»).

7. «Функціональне та нефункціональне тестування».

Методологія функціонального тестування відноситься до групи дій, які зосереджені на перевірці окремих функцій застосунка або функцій коду. Необхідні функції знаходяться в проєктній документації, незважаючи на те, що у деяких специфічних методологіях розробки, функціональні процеси базуються на основі випадків використання [166]. Переважна більшість функціональних тестів, відповідає на специфічні питання:

- «чи має користувач права щось робити?»;
- «чи працює взагалі ця окрема функція?».

Методологія нефункціонального тестування відноситься до аспектів програмного забезпечення, які не пов’язані із якоюсь певною функцією чи конкретною дією користувача. До таких аспектів відносяться:

- масштабованість чи інший критерій продуктивності;
- поведінка за умови певних обмежень;
- функції безпеки.

Тестування визначає конкретну частку програмного кода, в якій крайні межі масштабованості або продуктивності призводять до нестабільного виконання. Нефункціональні вимоги відображають якість

продукту, особливо в контексті перспективи придатності його для кінцевих користувачів.

8. У процесі **безперервного тестування** передбачається автоматизоване виконання, так званої «частини цілого конвеєра», щодо впровадження програмного засобу для отримання негайних відгуків відносно очікуваних бізнес-ризиків, напряму пов’язаних із кандидатом на випуск програмного забезпечення [151]. Безперервне тестування перевіряє як функціональні вимоги, так і відповідні нефункціональні вимоги. Загальний обсяг цього процесу тестування поширюється від етапу перевірки висхідних вимог, які називаються «історії користувачів», до етапу оцінки системних вимог, пов’язаних із головними бізнес-цілями.

9. В процесі проведення **деструктивного тестування,** тестувальник намагається власноруч спричинити збій програмного застосунка в цілому або його підсистеми. За цієї методології тестування перевіряє наскільки чи програмне забезпечення функціонує належним чином, навіть у тих випадках отримання застосунком недійсних або несподіваних вхідних даних. Цей процес дозволяє встановити надійність процедур перевірки вхідних даних і керування помилками [152].

Ін’єкція помилок програмного забезпечення — приклад «тестування на відмову».

10. Тестування продуктивності програмного забезпечення виконується для визначення роботи системи або підсистеми з точки зору стабільності за певного робочого навантаження.

Тестування продуктивності, спрямоване на процеси дослідження, вимірювання та перевірки наступних характеристик якості:

- надійність компонент системи;
- масштабованість системи;
- розподілення та використання ресурсів.

Навантажувальне тестування, в першу чергу, пов’язане із перевіркою того, чи продовжить система нормально функціонувати під

певним навантаженням, з великою кількістю інформації або з великою кількістю користувачів. Ця характеристика має назву «масштабованість» [148].

Тестування на витривалість — тестування навантаження системи, яке виконується у вигляді нефункціональної діяльності.

Об’ємне тестування — це спосіб перевірки функції програмного забезпечення, навіть, у тих випадках, якщо відповідні окремі компоненти (наприклад, текстовий файл або файл бази даних) непередбачувано збільшуються в розмірі протягом невеликого проміжку часу.

Стрес-тестування — це спосіб перевірки надійності під час неочікуваних або рідкісних навантажень.

Тестування стабільності (інша назва: «тестування на довговічність») — це спосіб перевірки спроможності програмного засобу безперервно працювати належним чином протягом прийнятного періоду.

На програмні системи реального часу накладаються суворі часові обмеження. Для перевірки дотримання часових обмежень, використовується тестування в реальному часі.

11. Тестування зручності використання зосереджується на перевірці, того чи є інтерфейс користувача простим у використанні і зрозумілим. Цей вид тестування не може бути автоматизованим та вимагає роботи реальних людей-користувачів, які контролюються кваліфікованими дизайнерами інтерфейсу користувача [150].

12. Тестування доступності проводиться для переконання у доступності програмного забезпечення для людей з обмеженими можливостями. Деякі з поширених тестових випадків доступності включають:

– колірний контраст між шрифтом і фоновим кольором є відповідним;

– розмір шрифту;

– альтернативні тексти для мультимедійного вмісту;

– можливість використання системи за допомогою клавіатури комп’ютера на додатків до миші.

13. Найважливіше значення для програмного забезпечення, яке оброблює конфіденційні дані, для запобігання вторгненню у систему зі сторони зловмисників має **тестування безпеки**.

14. Ціль **«тестування на інтернаціоналізацію та локалізацію»** - підтвердження, того що програмний засіб використовує різні мови та те що перебування у різноманітних географічних областях не впливає на коректність його роботи. Для перевірки цього атрибуту використовується механізм «псевдо локалізація». «**Тестування на інтернаціоналізацію та локалізацію»** перевіряє рівень адаптованості програмного засобу до різноманітних культур (наприклад, до використання валют різних країн або перебування у різних часових поясах). Також проводиться перевірка фактичного перекладу на різні людські мови [152].

Можливі збої локалізації та глобалізації включають:

– У випадку, якщо програмний продукт локалізується звичайним перекладом списку рядків, незважаючи на контекст, то існує вірогідність, що перекладач вибере хибний переклад для вихідного рядка, який складається зі слів, що можуть мати кілька значень.

– Технічна термінологія може стати неузгодженою, якщо над перекладом проєкта працюють кілька людей, не координуючи свої дії належним чином або діючи необережно.

– Дослівні переклади можуть здаватися невідповідними, штучними або надто технічними цільовою мовою.

– Програмісти можуть випадково залишити у вихідному коді, жорстко закодовані повідомлення мовою оригіналу без належного перекладу.

– Під час виконання окремі повідомлення могли бути створені автоматично, що призводить до граматично невірного вигляду кінцевого

рядка, або неправильного з функціональної точки зору, що може ввести в оману або заплутати кінцевих користувачів.

– У програмному засобі можуть використовуватись такі комбінації клавіш, які не прив’язані до жодної функції на клавіатурній розкладці вихідної мови, але які вимагає розкладка цільової мови.

– Програмне забезпечення може не підтримувати кодування символів цільової мови.

– Деякі шрифти та їх зображення, що відповідають вихідній мові, можуть не відповідати цільовій мови.

– Переклад певного рядка цільовою мовою може виявитись більшим за розміром, ніж програмний засіб здатен оброблювати. У такому випадку рядок може стати частково прихованим від користувача або призвести до збою програмного забезпечення.

– Програмне забезпечення може не надавати належної підтримки для того, щоб читати або вводити двонаправлений текст.

– Замість перекладу цільовою мовою програмний засіб може відобразити некоректне зображення у разі відсутності певної локалізації.

– При локалізації операційних систем може статися випадок, у якому системні файли конфігурації та змінні середовища матимуть різні імена, а також різні формати для дати та валюти.

15. Процес розробки програмних засобів, який передбачає використання широкого спектра різноманітних стратегій щодо запобігання та своєчасного виявлення дефектів для зменшення ризиків розробки програмного забезпечення називається **тестуванням розробки**. Цей процес виконується розробником або інженером програмного забезпечення на етапі будівництва «життєвого циклу розробки програмного забезпечення» [151]. При застосуванні методології тестування розробки, мета команди — усунення помилок конструкції до передачі програмного засобу на інший етап тестування. Ця стратегія, спрямована на підвищення якості кінцевого програмного забезпечення, а

також ефективності процесу розробки в цілому. В залежності від очікувань організації щодо протікання процесу розробки програмного засобу, тестування може включати різні технології статичного аналізу висхідного коду, методи та засоби аналізу інформаційного потоку, аналіз показників, однорангові перевірки коду, модульне тестування, аналіз охоплення коду, відстеження та інші методи тестування програмного забезпечення.

16. A/B-тестування – метод підготовки та виконання контрольованого експерименту, який спрямований на визначення чи зміни, які пропонується ввести, що призведуть до збільшення ефективності роботи чи навпаки. У цьому випадку клієнти періодично направляються спочатку до поточної (контрольної) версії функції, після чого до модифікованої версії [154]. Результати збираються для визначення кращої версії для досягнення бажаного результату.

17. Паралельне або одночасне тестування надає оцінку продуктивності та поведінки програмної системи, яка використовує паралельні алгоритми за будь-яких умов. Цей тип тестування виявляє такі "типові" проблеми, як взаємоблокування, умови змагання та проблеми з обробкою спільної пам'яті/ресурсу [155].

18. Мета **перевірки на відповідність** — перевірка того, що програмний продукт працює відповідно до визначених стандартів. Наприклад: компілятори перевіряються із особливою ретельністю для визначення відповідності визнаному стандарту цільової мови програмування [156].

19. Порівняльне тестування виходу (інша назва – «тестування моментальних знімків») – створення очікуваного результату відображення для:

- порівняння текстових даних;
- порівняння знімків екрана інтерфейсу користувача.

Цей підхід, на відміну від інших підходів та технологій тестування, не виявляє помилки автоматично, а натомість надає звіт-результат, який має бути оціненний людиною на наявність невідповідностей [157].

20. Тестування властивостей — техніка тестування, яка вимагає ствердження того, що при конкретних вхідних даних будуть отримані відповідні очікувані результати. Тестувальник випадковим чином створює велику кількість даних для перевірки, потім проводить запуск застосунка на кожному із них і перевіряє істинність обраної конкретної «властивості», яка має бути істинною для кожної пари входу та виходу. Наприклад: кожні вихідні дані із функції серіалізації мають бути прийняті відповідною зворотною функцією десеріалізації, а кожні вихідні дані із функції сортування мають бути монотонно зростаючим списком, що містить точно ті самі елементи, що й вхідні дані [158-159].

Програмні бібліотеки тестування, які підтримують дану технологію тестування, дозволяють тестувальнику повністю контролювати ту стратегію, яка будує випадкові вхідні дані для забезпечення повного охоплення особливих випадків або тих випадків вхідних даних у яких присутні певні шаблони, необхідні для повної перевірки аспектів реалізації, що тестується.

21. Метаморфічне тестування — це метод тестування програмного забезпечення на основі властивостей, який є ефективним підходом для вирішення «проблеми тестового оракула» та «проблеми створення тестового випадку». «Проблема тестового оракула» полягає в складності визначення очікуваних результатів вибраних тестів або визначення того, чи узгоджуються фактичні результати з очікуваними результатами [160-164].

Процес тестування програмного забезпечення спричиняє виникнення певної кількості «артефактів». Фактично вироблені «артефакти» — чинники поточної моделі розробки програмного

забезпечення, зацікавлених сторін і організаційних потреб. До таких «артефактів» відносяться:

- **План тестування** — це документ, у якому детально описується та стратегія, яка буде застосована до усіх заходів тестування, присутніх у плані [165-169]. План включає наступні атрибути:

- цілі тестування;
- обсяг;
- процеси та процедури;
- вимоги до персоналу;
- плани на випадок непередбачених обставин.

План тестування представляється у вигляді єдиного документа, який містить усі можливі типи тестування (наприклад, план приймання або системного тестування) і окремі думки відносно процесу планування розробки або план тестування видається у вигляді загального плану тестування, який включає огляди більш ніж одного процесу детального тестування (плану) [170]. У деяких випадках тестовий план входить до значно ширшої «стратегії тестування», яка документує загальні підходи організації процесу тестування.

- У розробці програмного забезпечення **матриця відстеження** - це документ у формі таблиці, який використовується для визначення повноти зв’язку функціональних вимог шляхом співвіднесення будь-яких двох базових документів за допомогою відношення багато-до-багатьох. Матриця відстеження використовується із вимогами високого рівня (наприклад: маркетингових вимог) і детальними вимогами застосунка до відповідних частин проєктування високого рівня, детального проєктування, плану тестування та тестових випадків [171].

- **Тестовий випадок** має наступну структуру:

- унікальний ідентифікатор;
- посилання на вимоги зі специфікації проєкту;
- попередні умови;

- події;
- серії кроків, які також називають «дії», що мають бути виконаними;
- операція введення;
- операція виведення;
- порівняння очікуваного результату та фактичного результату.

Тестовий випадок — це вхідні дані та очікуваний результат. Найпростіший тестовий випадок має вигляд: «для умови х ваш отриманий результат є у», хоча в більшості випадків тестові випадки описують більш детально вхідний сценарій і очікувані результати.

Тестові випадки описують послідовності різноманітних кроків (проте у більшості випадків такі кроки групують в окрему процедуру, яка застосовується разом із кількома тестовими випадками з міркувань оптимізації), які вони завжди мають один очікуваний результат або групу результатів [172].

До додаткових характеристик тестових випадків відносяться:

- конкретний крок тесту або порядковий номер виконання;
- унікальний ідентифікатор випадку;
- пов’язані вимоги;
- категорія тесту;
- глибина тестів;
- прапорці, які є сигналами того, чи є тест автоматизованим і чи був він автоматизований.

Більші тестові випадки також містять попередні стани або кроки та описи. Тест також включає відповідне місце, яке зберігає фактичний результат. Кроки тестів зберігаються у вигляді:

- звичайного текстового файлу;
- електронної таблиці;
- об’єкта бази даних.

Система керування базами даних також надає можливість перегляду результатів попередніх тестів, джерела генерації результатів та

налаштування системи, які були використані для генерації кінцевих результатів [173].

- **Тестовий сценарій** — процедура або програмний код, який повністю імітує дії користувача. Цей термін є похідним від результату роботи, створеного автоматизованими інструментами регресійного тестування.
- У розробці програмного забезпечення **набір тестів** (інша назва – «набір перевірки») — це набір тестів, які призначені для тестування програмного забезпечення та для демонстрації відповідного набору поведінки. Набір тестів містить докладні інструкції або цілі для кожної колекції тестів та базується на налаштуваннях тієї системи, що буде основною під час процесу тестування. До груп тестів включаються попередні стани або кроки та описи наступних тестів [174-175].
- **Тестовий пристрій або «тестові дані».** Кілька наборів значень або даних у більшості випадків використовуються для перевірки однієї частини певної функції. Усі тестові значення та змінні компоненти середовища збираються в окремих файлах [176].

2.3. Використання технології блокчейн в проєкті інформаційно-аналітичної системи

Технологія блокчейн [134] здатна зробити революцію в індустрії електронного навчання, підвищивши рівень безпеки, довіри та співпраці, шляхом створення нової бізнес-моделі та надання студентам кращого доступу до якісної освіти таблиця 2.9.

Таблиця 2.9.

Якісні зміни використання технології блокчейн для сучасних освітніх середовищ

Покращена безпека	Децентралізована природа технології блокчейн покращує безпеку платформ електронного навчання

	[135-136]. Технологія блокчейн забезпечує безпечне зберігання та перевірку даних студентів, сертифікатів і облікових даних, зменшуючи ризик шахрайства та витоку даних
Автентифікація та довіра	Блокчейн забезпечує надійну перевірку навичок і досягнень студентів за допомогою децентралізованих рішень цифрової ідентифікації. Це гарантує, що облікові дані та сертифікати захищені від підробки та ефективно перевіряються установами та роботодавцями, що підвищує довіру до онлайн-навчання [136].
P2P навчання та обмін матеріалами	Блокчейн сприятиме поступовому навчанню, шляхом створення платформ, де студенти можуть безпосередньо ділитися знаннями, контентом і ресурсами без посередників. Цей децентралізований підхід заохочує до співпраці, розширює доступ до якісних навчальних матеріалів і зменшує витрати [137].
Мікроплатежі та захист авторських прав	Завдяки блокчейну навчальні платформи отримують можливості для дослідження моделі мікроплатежів, коли студенти платять за певні модулі, заняття чи оцінювання. Це сприяє створенню високоякісного освітнього контенту та захисту авторських прав творців за допомогою технології смарт-контрактів і міток часу.
Прозоре ведення документації	Технологія розподіленої книги Blockchain безпечно зберігає та відстежує освітні записи, включаючи сертифікати про закінчення, ступені та кваліфікацію

	[138]. Це забезпечує прозору систему ведення записів, стійку до змін і дозволяє перевірку облікових даних роботодавцями.
Аналіз даних і персоналізація	Безпечно збираючи та аналізуючи дані студентів у блокчейні, навчальні платформи отримують уявлення про поведінку, уподобання та моделі навчання здобувачів освіти. Ці дані використовуються для персоналізації навчального досвіду, рекомендації відповідного вмісту і покращення загальних результатів навчання.

Одним із застосувань технології блокчейну в електронному навчанні є перевірка облікових даних. Завдяки блокчейну заклади освіти отримують можливість видачі цифрових сертифікатів або дипломів, які надійно записуються в блокчейні [139]. Це усуває потребу в процесах ручної перевірки, що полегшує роботодавцям або іншим закладам освіти процеси верифікації автентичності облікових даних. Крім того, блокчейн технології розширюють можливості студентів, надаючи їм права власності і контроль над своїми навчальними документами. Здобувачі вищої освіти отримують цифровий гаманець у блокчейні, що зберігає їхні досягнення, сертифікати чи документи, що підтверджують конкретні навички. Це забезпечує прозорий і незмінний запис «освітньої подорожі», що дозволяє студентам ділитися своїми обліковими даними з потенційними роботодавцями чи академічними установами [140].

Інша розробка програми блокчейну в електронному навчанні стосується обміну матеріалами і захисту авторських прав. Blockchain створює децентралізовану платформу, завдяки котрій науково-педагогічні працівники отримують можливість безпечно ділитися навчальним контентом, забезпечуючи належне посилання та захист авторських прав [141-143]. Це сприяє співпраці та інноваціям у навчальній спільноті, а

також забезпечує авторам навчальних матеріалів справедливу винагороду за роботу.

Потенціально ефективне поєднання блокчейну та електронного навчання зробить революцію в індустрії освіти. Технологія блокчейн, відома своєю децентралізованою та безпечною природою, підвищить прозорість, ефективність і надійність платформ електронного навчання.

Використовуючи блокчейн, платформи електронного навчання отримують можливість надавати захищені від підробки записи про досягнення студентів, сертифікати та кваліфікації. Ці записи зберігаються в розподіленій «книзі», що гарантує неможливість їх змін або фальсифікації. Це забезпечує роботодавцям і закладам освіти впевненість у достовірності освітніх досягнень особи, шляхом створення надійнішої та перевіреної системи [144].

Крім того, блокчейн надає можливість ввімкнення мікроплатежів та стимулювання студентів за допомогою токенізованих винагород, створюючи більш привабливе та мотивуюче освітнє середовище. Учні отримують токени або цифрові активи за проходження курсів, досягнення або демонстрацію своїх знань у певних областях. Ці винагороди можуть бути використані для подальшого навчання, обміну на інші товари та послуги або навіть зберігання у вигляді інвестицій.

Крім того, блокчейн сприяє одноранговій взаємодії в рамках платформ електронного навчання. Усуваючи посередників, блокчейн забезпечує пряме спілкування, співпрацю та обмін знаннями між студентами та викладачами. Цей децентралізований підхід заохочує відчуття спільності та розширення можливостей, сприяючи більш інклюзивній та доступній екосистемі електронного навчання [145].

Крім того, незмінність і прозорість блокчейна здатна вирішити такі проблеми, як плагіат і крадіжка інтелектуальної власності. «Розумні контракти» можуть бути використані для захисту авторських прав і

справедливої винагороди для авторів контенту, заохочуючи виробництво високоякісних навчальних матеріалів.

Використання технології блокчейн в освіті має великий потенціал у майбутньому [146-147]. Як вже зазначалось, технологія блокчейн пропонує різні переваги:

- підвищена безпека;
- покращене керування даними;
- вдосконалений механізм перевірки академічних сертифікатів.

Можливі застосування наведено в таблиці 2.10.

Таблиця 2.10.

Сфери застосування технології блокчейн в освіті

Безпечне ведення записів	Блокчейн забезпечує незмінну та децентралізовану систему реєстру, яка може безпечно зберігати освітні записи, такі як ступені, сертифікати та дипломи. Це гарантує неможливість підробки або фальсифікації записів [146].
Перевірка облікових даних	За допомогою блокчейну заклади освіти можуть видавати цифрові сертифікати та дипломи, які перевіряються роботодавцями чи іншими закладами освіти.
Прозорі оцінювання	Блокчейн сприяє прозорому оцінюванню, яке може бути перевірене шляхом запису даних із мітками часу особи оцінювачів і критеріїв оцінювання. Це підвищує довіру до процесу оцінювання [145].
Одноранговий обмін обліковими даними	Блокчейн дозволяє студентам мати власність і контролювати свої навчальні записи. Вони отримують можливість ділитися своїми перевіреними обліковими даними з потенційними роботодавцями чи іншими

	установами безпечно, усуваючи потребу в посередниках.
Мікроакредитації та навчання впродовж життя	Блокчейн дозволяє створювати та розпізнавати мікро акредитацій, що дозволяє людям демонстрацію певних навичок та досягнень [147]. Це підтримує навчання впродовж життя, забезпечуючи детальніший і гнучкіший підхід до освіти та просування по службі.

Незважаючи на значний потенціал, широке впровадження блокчейну в освіті все ще вимагає подолання певних проблем:

- технічні бар'єри впровадження;
- встановлення галузевих стандартів;
- проблеми конфіденційності.

Тим не менш, майбутнє блокчейну в освіті виглядає багатообіцяючим для революції в процесі ведення записів, верифікації облікових даних і можливостей навчання.

Висновки до другого розділу

1. Архітектура нового віртуального освітнього середовища для закладів вищої освіти України – інтернет-застосунок, доступ до якого мають усі учасники освітнього процесу, незалежно від місць їх фізичного перебування та від апаратної платформи якими вони користуються.

2. Архітектурний шаблон, який забезпечує необхідну швидкість та гнучкість розробки головної частини інтернет-застосунка віртуального освітнього середовища, - це шаблон модель - вид - представлення.

3. При розробці віртуального освітнього середовища мають бути використані найсучасніші технології для побудови та подальшої оптимізації користувацької та серверної частин системи та відповідні формати даних.

4. При розробці віртуального освітнього середовища мають бути

задіяні механізми та технології забезпечення якості програмного забезпечення для організації ефективних процесів внесення змін у роботу системи та реагування на зміни в освітньому процесі протягом навчального року.

Для підвищення рівнів безпеки та співпраці в сучасній освіті у віртуальних освітніх середовищах та створення нової освітньої бізнес-моделі освіти можуть бути використані технології блокчейн.

РОЗДІЛ 3
ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ KSU24

3.1. Загальна архітектура інформаційно-аналітичної системи KSU24

Інформаційно-аналітична система KSU24 розробляється як клієнт-серверний застосунок згідно шаблону проєктування модель–шаблон–подання. Загальна архітектура, складові частини інформаційно-аналітичної системи KSU24, додаткові сторонні та вбудовані бібліотеки та сервіси, що забезпечують окремі задачі на зразок логування представлені на рис.3.1.

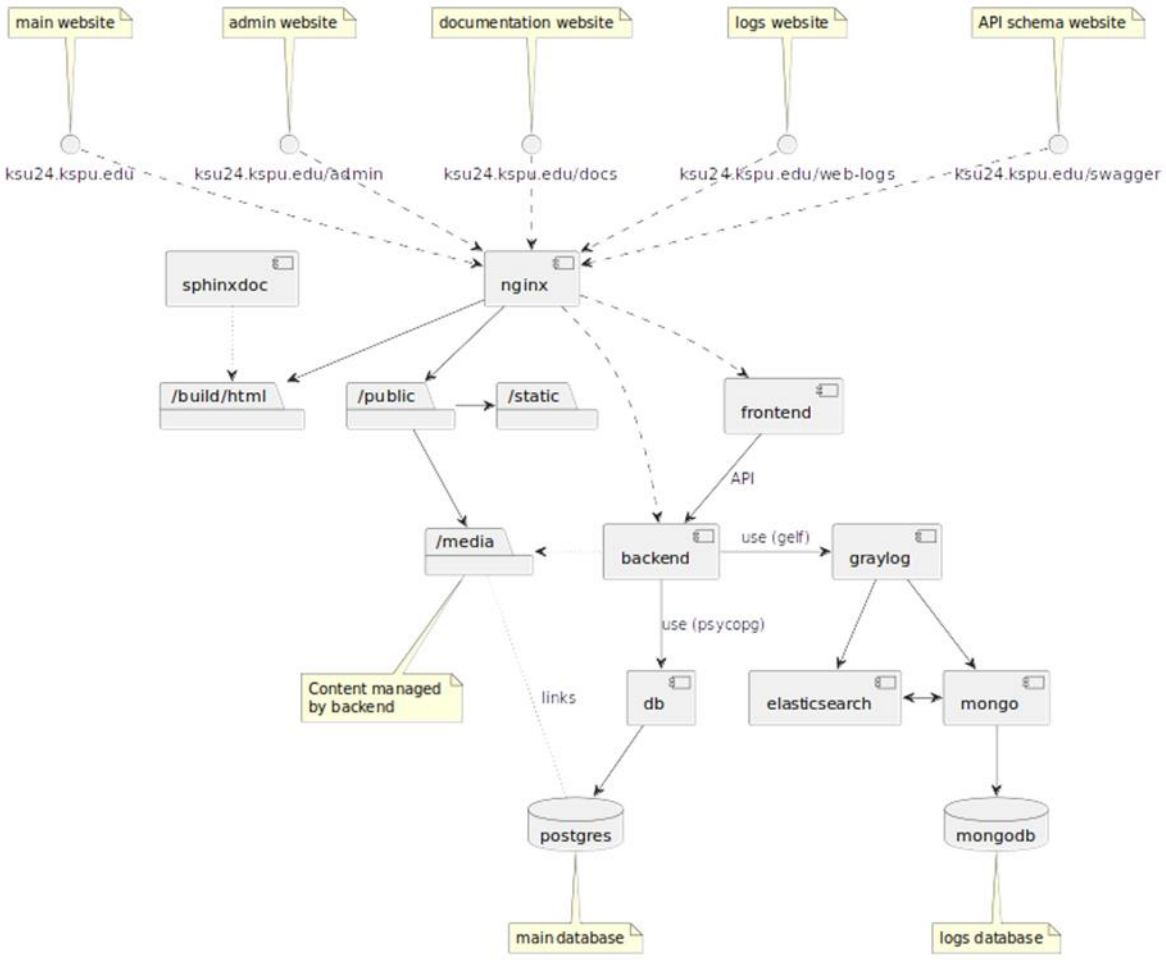


Рисунок 3.1. Архітектура інформаційно-аналітичної системи KSU24

На найнижчому рівні ієрархії складових частин знаходиться база даних. Уся інформація, що відноситься безпосередньо до функціонування закладу вищої освіти (наприклад, облікові записи студентів та науково-педагогічних працівників, навчальні плани, відомості, журнали, залікові книжки, фінансова інформація та ін.) зберігається в **основній базі даних,** що використовує реляційну базу даних postgreSQL.

Найголовнішою задачею цього головного сховища інформації є забезпечення надійності даних, що зберігаються, завдяки механізму транзакцій. Поряд із основною базою у проєкті застосовується окрема база для ведення журналу всіх подій, що відбуваються в системі. Вона вже використовує noSQL базу даних mongoDB, яка вже не містить транзакцій, тому не вважається настільки ж надійним сховищем даних, проте у порівнянні із реляційними базами даних, надає дуже ефективні інструменти пошуку одного необхідного кортежу даних у великій кількості записів, що є дуже важливим при роботі із журналом подій системи, для того надати змогу, насамперед розробникам, якомога швидше реагувати на можливі збої в роботі системи. Як видно із схеми, база даних для логування подій використовується модулем GrayLog: програмне забезпечення для керування журналами та аналітики безпеки, що використовує «elastic search», який забезпечує швидкий пошук, має близький до реального часу пошук і підтримує багато категорій пошуку.

Модулі, що реалізують бізнес-логіку, а саме серверна та клієнтська частини застосунку, позначені як backend та frontend відповідно. Серверна частина розроблена на базі фреймворка DJango версії 4.2.7, для мови програмування Python версії 3.10.12.

Інструменти та номенклатура фреймворку Django використовуються для побудови основної бізнес-логіки інформаційно-аналітичної системи KSU24. Django це безкоштовна вебплатформа з відкритим кодом на основі Python — яка працює на вебсервері. Вона дотримується архітектурного шаблону модель–шаблон–подання (MTV).

Головна ціль використання Django — спрощення процесу розробки «складних» вебсайтів, які використовують бази даних для зберігання та обробки інформації. Використання фреймворка вимагає багаторазового використання та «приєднуваності» компонентів, меншої кількості коду, низького зв'язку, дозволяє швидкий розвиток проєкту та впроваджує принцип «не повторюйся». Python використовується всюди, навіть, для налаштувань, файлів і моделей даних. Django також автоматично надає додатковий адміністративний інтерфейс для базових задач створення контенту, читання контенту, оновлення та видалення контенту. Цей інтерфейс генерується динамічним шляхом самоаналізу та налаштовується за допомогою моделей адміністратора.

Незважаючи на наявність власної номенклатури, наприклад тієї, яка використовується для привласнення імен об'єктам, що викликаються для генерації «відображень» HTTP-відповідей, базову структуру Django можна розглядати як архітектуру MVC. До її складу входять інструмент «об'єктно-реляційного відображення» (ORM), який є посередником між моделями даних, визначеними як класи Python і реляційним сховищем даних («Модель»), системою, що обробляє HTTP-запити із системою вебшаблонів («Перегляд») і диспетчером URL-адрес на основі регулярного виразу («Контролер»).

До основних компонент Django, так званого «ядра», входять:

– легкий автономний вебсервер для розробки та тестування;

– механізм для процесів «серіалізації» та перевірки форм, який може перекладати форми HTML і перетворювати отримані значення у вигляд, який дозволить зберегти їх у базі даних;

– система шаблонів, яка використовує концепцію успадкування, запозичену з об'єктно-орієнтованого програмування;

– система для організації механізму «кеш», який є спроможним користуватися будь-яким із кількох доступних методів кешування;

– підтримка класів проміжного ПЗ, які можуть бути задіяні при обробці запитів, на будь-якому з етапів цієї обробки і виконувати спеціальні функції;

– внутрішня диспетчерська система, яка дозволяє компонентам програми реалізовувати обмін подіями один від одного за допомогою попередньо визначених сигналів;

– система інтернаціоналізації до складу якої входять переклади різними мовами вбудованих компонентів Django;

– система серіалізації, яка може створювати та обробляти формати даних у вигляді XML-файлів та/або JSON-файлів, всередині екземплярів моделі Django;

– система розширення можливостей механізму шаблонів;

– інтерфейс до вбудованої інфраструктури модульного тестування Python.

Також разом із основним ядром фреймворку Django поставляються наступні додаткові застосунки, які представляють:

– розширювану систему автентифікації;

– динамічний адміністративний інтерфейс;

– інструменти для створення каналів новин RSS і Atom;

– фреймворк «Сайти», який надає можливість запускати кілька вебсайтів на одному «ядрі» Django, кожен із власним вмістом і програмами;

– функції для роботи із сервісом Google Sitemap;

– вбудований захист від підробки міжсайтових запитів, міжсайтових сценаріїв, упровадження SQL, злому паролів та інших типових вебатак, більшість із яких увімкнено за замовчуванням;

– фреймворк для створення ГІС-додатків.

Клієнтська частина інформаційно-аналітичної системи KSU24 використовує технологію ReactJS — безкоштовну java-script бібліотеку для створення користувацьких інтерфейсів. React можна використовувати

для розробки односторінкових, мобільних або серверних застосунків. Оскільки React займається лише інтерфейсом користувача та відтворенням компонентів у DOM, застосунки React часто покладаються на бібліотеки для маршрутизації та інших функцій на стороні клієнта. Ключовою перевагою React є те, що він повторно відтворює лише ті частини сторінки, які змінилися, уникаючи непотрібного повторного відтворення незмінених елементів DOM. Частина інформаційно-аналітичної системи KSU24, з якою взаємодіють здобувачі вищої освіти та більшість науково-педагогічних працівників, побудована цілком за допомогою бібліотеки React, відмінно від адміністративної частини KSU24, що разом із бібліотекою React використовують також фреймворк Node.js для зв'язку із серверною частиною для забезпечення процесу, так званого, «серверного рендерингу».

Рештою складових частин, що відносяться до архітектури застосунку, є вебсервер Nginx — програмне забезпечення та базове обладнання, яке приймає запити через HTTP (мережевий протокол, створений для розповсюдження вебвмісту) або його безпечний варіант HTTPS, генератор документації Sphinx та ще декілька допоміжних вебзастосунків, задачею яких є насамперед підтримка та оптимізація процесу розробки. Серед них слід виділити:

– сервіс swagger, що являє собою автоматично згенеровану документацію функцій серверною частини, яка є дуже важливою для програмістів, які працюють із клієнтською частиною KSU24;

– сервіс для роботи із системними журналами;

– сервіс для документації проєкту.

3.2. Структура класів та головні елементи серверної частини інформаційно-аналітичної системи KSU24

Як і в будь-якій системі керування взаємовідносинами, найбільш складну та розгалужену структуру в інформаційно-аналітичній системі

KSU24 має частина, що впроваджує функціональність та взаємовідносини людських ресурсів університету. В основі усієї цієї частини знаходиться клас «Особа» (рис. 3.2), який містить базову інформацію із можливостями редагування за умови наявності відповідного рівня дозволу про одну певну людину. Окремо одразу слід відмітити 3 обов'язкових поля, які будуть присутні у більшості класів системи:

– *created* — дата та час, коли запис про особу було створено;

– *modified* — дата та час, коли запис про особу було змінено;

– *deleted* — дата та час, коли запис про особу перестав бути активним у системі.

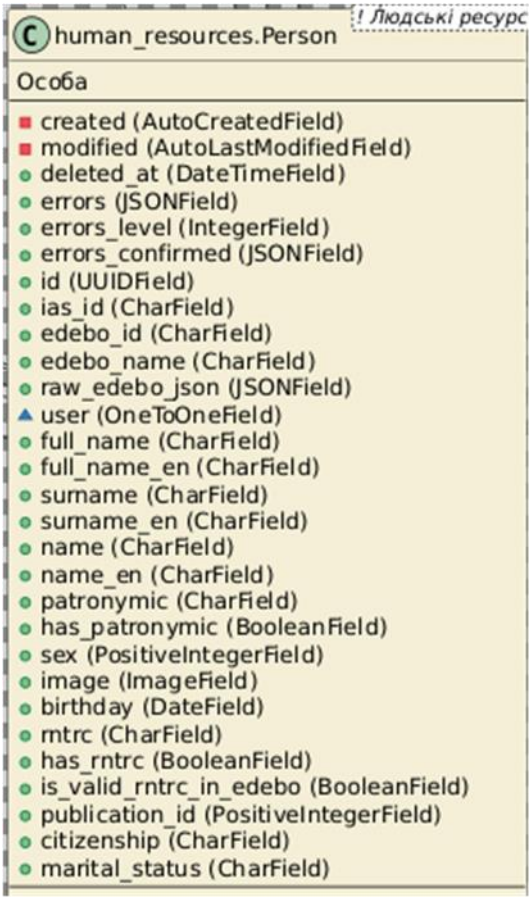


Рисунок 3.2. Структура класу «Особа»

Інші поля містять як інформацію про конкретну людину (ім'я, прізвище українською та англійською мовами, дата народження, стать, громадянство, сімейний стан), так і технічну інформацію, яка необхідна

як Херсонському державному університету, так і МОН України: ідентифікатор та ім’я в Єдиній державній електронній базі з питань освіти (скор. ЄДЕБО), ідентифікатор у системі ІАС ХДУ.

Одразу після того, коли здобувач вищої освіти реєструється у системі, для нього створюється запис як про особу у базі даних, з тією інформацією, яка доступна на момент реєстрації. Частину інформації користувач заповнює власноручно, а частину (наприклад, ідентифікатор у ЄДЕБО) автоматично заповнює система, здійснюючи запит до відповідних сервісів. Також людині буде потрібно ввести в особистому кабінеті додаткову інформацію про себе, яка буде необхідна для успішної роботи або навчання в університеті, котра теж представлена відповідним набором класів (рис. 3.3).

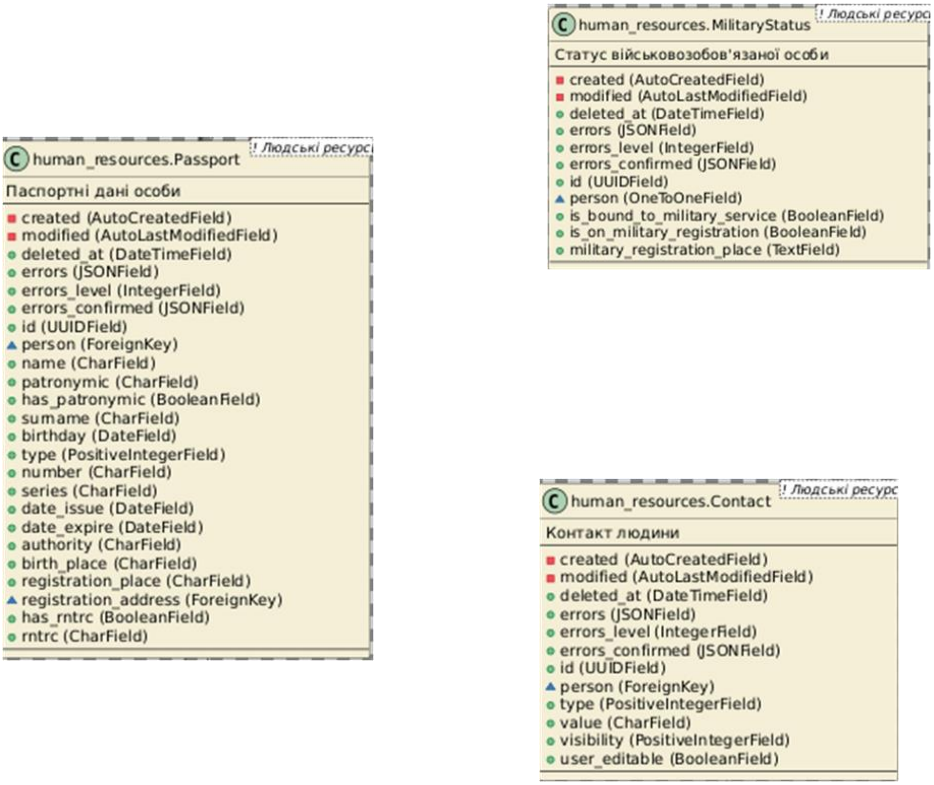


Рисунок 3.3. Класи, що відповідають за розширену інформацію про людину в інформаційно-аналітичній системі KSU24.

До цієї інформації відносяться: дані про військовий облік (клас MilitaryStatus) якщо це чоловік, паспортні дані (клас Passport) та контакти людини (клас Contact). Якщо реєструється здобувач, він також має заповнити публічну оферту (рис. 3.4).

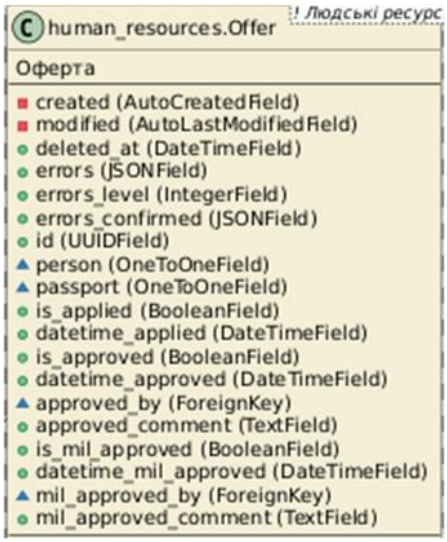


Рисунок 3.4. Структура класу публічної оферти здобувача

Після фізичного підписання здобувачем оферти надання усіх необхідних документів до приймальної комісії та успішного складання вступних іспитів він отримує статус здобувача вищої освіти та відповідну роль у системі KSU24, що теж представлено відповідним класом (рис. 3.5).

Цей клас відповідає за зберігання та обробку усієї інформацію, що відноситься до студента Херсонського державного університету, зв’язок із обраною спеціальністю, академічною групою, рівнем освіти, усією інформацією про нього як студента з ЄДЕБО, курс, форма навчання, роки вступу/завершення, інформація про диплом з освіти, який здобувач вищої освіти вже має на момент вступу та ін.

З іншого боку, у випадку, коли у системі реєструється науково-педагогічний працівник, то для нього теж створюється запис у базі даних, згідно до класу «Особа» (рис. 3.2), однак він отримує роль, функціонал якої впроваджує клас «Працівник» (рис. 3.6).

Рисунок 3.5. Структура класу студента

Цей клас також містить усю персональну інформацію про співробітника, як у випадку зі здобувачем вищої освіти (адреса, паспортні дані тощо), і, крім того, впроваджує інформацію про факультет/кафедру/відділ, де працює співробітник, його науковий ступінь, стаж роботи та ін.

Окремо варто зазначити, що завдяки реалізованій структурі, людина, маючи один обліковий запис у системі KSU24, може мати безліч ролей: одночасно працювати викладачем на одному факультеті, здобувати другу освіту у якості здобувача вищої освіти на іншому (або навчатися в

аспірантурі) та виконувати роль керівника/співробітника одного із відділів університету.

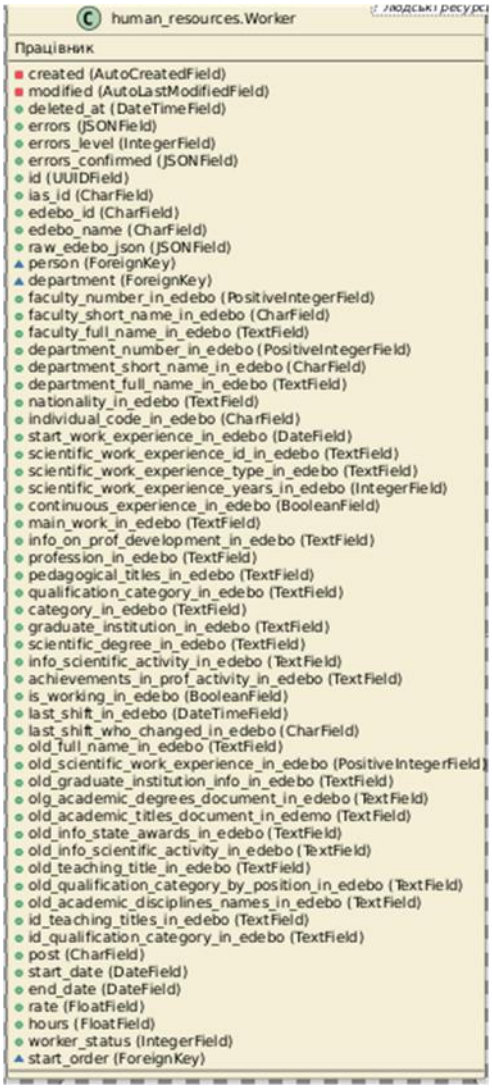


Рисунок 3.6. Структура класу «Працівник» для співробітників

Робота із документами, які необхідні для здобувачів вищої освіти та співробітників протягом усього процесу навчання в університеті, забезпечується відповідними класами (рис. 3.7).

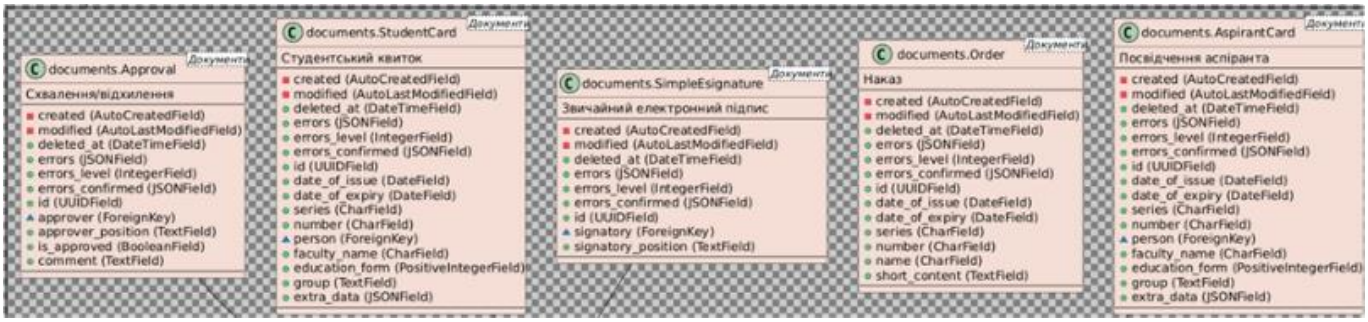


Рисунок 3.7. Документи в системі KSU24

Підтримка безпосередньо самого навчального процесу реалізована відповідно класами для робочих програм із дисциплін, власне дисциплін, академічних журналів, відомостей, залікових книжок та класами чатів, опитувань та ін., що забезпечують зворотній зв'язок від здобувачів вищої освіти після проведення занять, академічних сесій.

Навчальні дисципліни в KSU24 представлені наступними класами (рис. 3.8):

- *навчальний предмет* (англ. Subject), що відповідає за обробку та зберігання основної інформації про дисципліну (її вміст);
- «*ревізія навчального предмету*» (англ. SubjectInEpRevision), основна задача якого у керуванні змінами у плані навчального предмета, які можуть трапитися у новому навчальному році порівняно з минулими.

Наведемо приклад: одного року на предмет «Теорія баз даних» відводилось 90 годин навантаження, а в іншому на цей предмет планується вже 120 годин; або взагалі в навчальному плані предмет може змінити назву. Цей клас відстежує усі зміни, забезпечує зв'язок між різними версіями навчальних дисциплін, зв'язок безпосередньо з навчальною програмою, з академічним журналом.

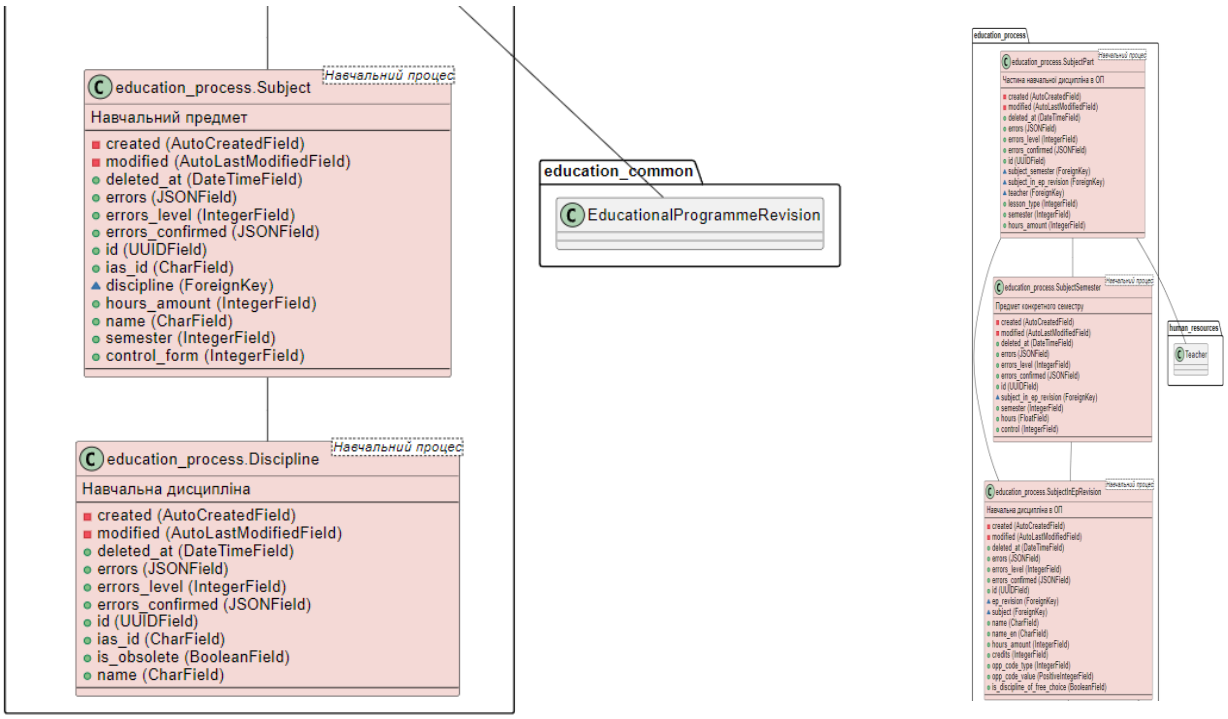


Рисунок 3.8. Структура класів для представлення навчальних дисциплін

- класи «предмет конкретного семестру» та «частина навчальної дисципліни в ОП» забезпечують роботу з інформацією, що актуальна для конкретної дисципліни саме для обраного семестру: кількість годин, викладач, який веде предмет, форма контролю.

На основі навчальних дисциплін, що викладаються в кожному семестрі, формуються академічні журнали, які призначені для безпосереднього ведення навчального процесу. Журнал створюється з конкретного предмету на конкретний семестр, для конкретної групи студентів та конкретного викладача. Слід зазначити, що журнали для студентів і науково-педагогічних працівників відрізняються, як за виглядом, так і за функціоналом. Здобувач вищої освіти може переглядати лише власний журнал з усіх предметів та бачить лише свої академічні здобутки (якщо він не має ролі «староста»), викладач бачить журнал для усієї групи та має доступ до значно ширшого набору функцій: виставлення/редагування оцінок, закриття журналу зі своїм підписом, після чого до журналу вже не можна вносити зміни, проте стає можливим створення відомостей. Структура класів для академічних журналів представлена на рис. 3.9.

Власне за збереження, обробку основної інформації з ведення навчальної дисципліни (який саме науково педагогічний працівник веде цей журнал, з якої навчальної дисципліни, яка група, який предмет, кількість годин, який навчальний рік/семестр, форма контролю, кількість кредитів та зв’язок з опитуваннями) та допоміжної технічної інформації (який користувач створив даний журнал, який користувач може переглядати його, вносити до нього зміни) відповідає клас Gradebook.

Для збереження та обробки інформації, яка відноситься безпосередньо до процесу проведення навчального заняття, відповідає клас «Заняття» (англ. Lesson). Ця інформація включає:

- дата проведення заняття;
- посилання на відеоматеріали заняття;

- посилання на відповідні матеріали з сервісу KSUOnline;
- посилання на зустріч у сервісі Zoom;
- максимальна оцінка за предмет;
- посилання на опитування після заняття;
- проведення заняття в синхронному або асинхронному форматі.

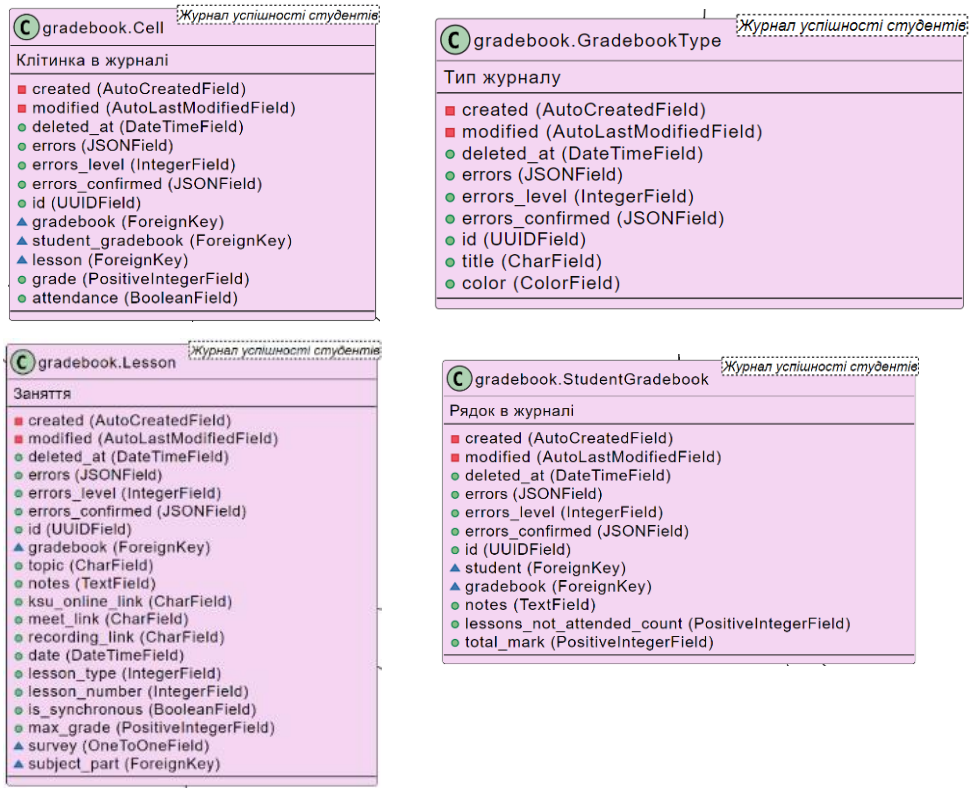


Рисунок 3.9. Структура класів академічного журналу

Для збереження та обробки інформації про те, який студент отримав яку оцінку за предмет, чи був студент присутній на занятті, відповідає клас «Клітинка в журналі» (англ. GradebookCell).

Після того, як усі навчальні заняття з предмету були проведені, зафіксовані в журналі та здобувачі вищої освіти отримали відповідні оцінки, викладач може «закрити» журнал. Після цього стає неможливим внесення змін до журналу та відкривається доступ до створення відомостей. Також слід окремо зазначити, що існує можливість «перевідкрити» журнал, якщо необхідно зробити незначні зміни

(виправити якусь технічну помилку). Структура відомості з навчальної дисципліни представлена на рис. 3.10.

Рисунок 3.10. Класи, що відносяться до відомостей з навчальних дисциплін

Після того, як викладач внесе усю відповідну інформацію у відомість, він підтверджує її своїм підписом, а також свої підписи ставлять члени комісії, за це відповідає клас «Підпис до відомості обліку успішності» (англ. RecordBookGroupSign). Останнім кроком є підтвердження відомості співробітниками деканату/навчального відділу. Цю функціональність забезпечує клас «Схвалення/відхилення відомості обліку успішності» (англ. RecordBookGroupApproval). Якщо певна кількість здобувачів вищої освіти потрапляє на перездачу або на комісію, то для цих випадків створюється окрема відомість, яку також необхідно буде окремо завірити та підписати як викладачу, так і членам комісії.

Класи навчальної відомості відповідають саме за адміністративну частину функціоналу, за обробку інформації про оцінки здобувачів вищої освіти, які вони отримали, пройшовши відповідну дисципліну, відповідають класи «Залікова книжка» та «Запис до залікової книжки» (рис. 3.11).

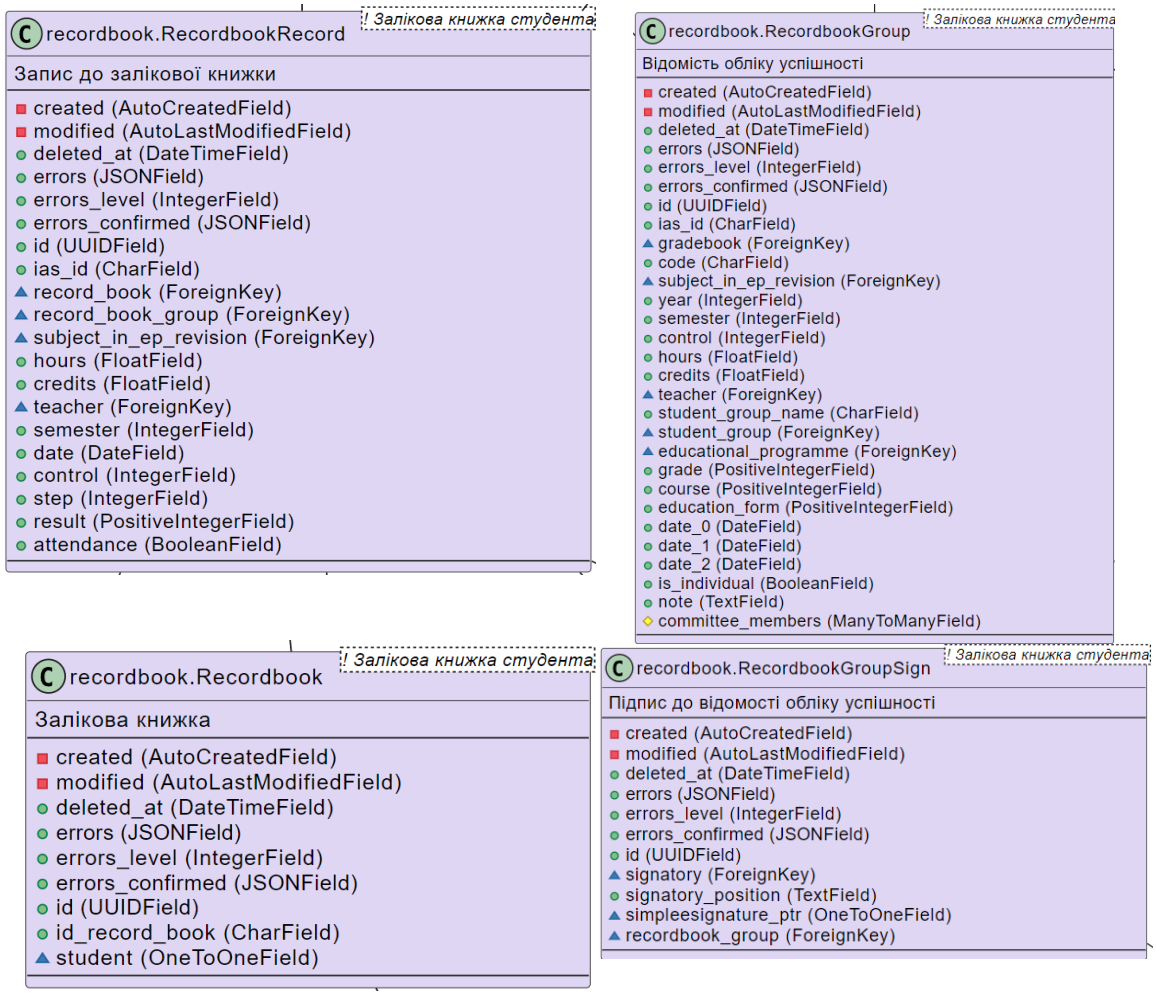


Рисунок 3.11. Структура класів залікових книжок студентів

Власне клас «Залікова книжка» забезпечує зв’язок між здобувачем вищої освіти та його оцінками з певного предмета, а клас «Запис до залікової книжки» безпосередньо всю необхідну інформацію про результати отримані здобувачем вищої освіти: яка оцінка, яка навчальна дисципліна, хто викладач, який семестр, яка форма контролю, кількість годин, кількість кредитів та дата виставлення оцінки.

Окремо від класів, що забезпечують процес виставлення оцінок із дисциплін, використовуючи академічні журнали та відомості, винесено у відповідний клас функціональність для оцінювання кваліфікаційних робіт здобувачів вищої освіти (рис. 3.12).

Цей клас містить інформацію про прізвище, ім’я та по-батькові керівника, посилання власне на саму роботу та тему кваліфікаційної роботи.

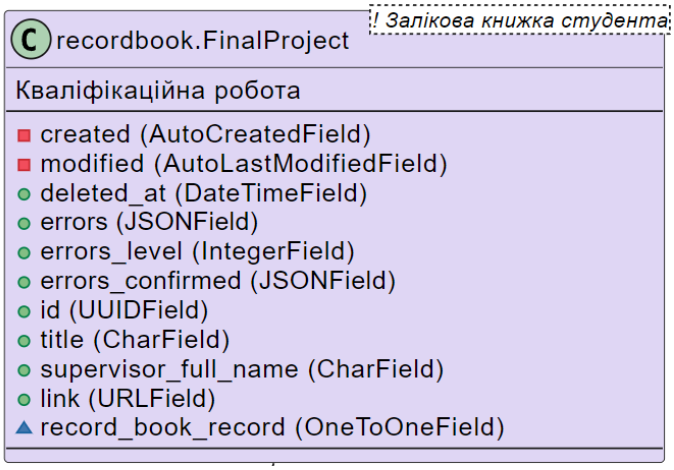


Рисунок 3.12. Структура класу для кваліфікаційної роботи студента

Ще однією важливою частиною забезпечення навчального процесу є функціональність для створення, редагування та перегляду навчального розкладу. Як і у випадку з журналом, розклад у KSU24 є персоналізованим. Здобувачі вищої освіти та науково-педагогічні працівники бачать лише власний розклад занять. Розклад також напряму пов'язаний із факультетом та враховує аудиторії, де може бути проведене заняття, у випадку, якщо заняття проводиться не в дистанційному форматі. Розклад формується на початку навчального семестру на основі навчального навантаження та дисциплін, які будуть викладатися. Загальна структура класів, що відповідають за представлення навчального розкладу презентована на рис. 3.13.

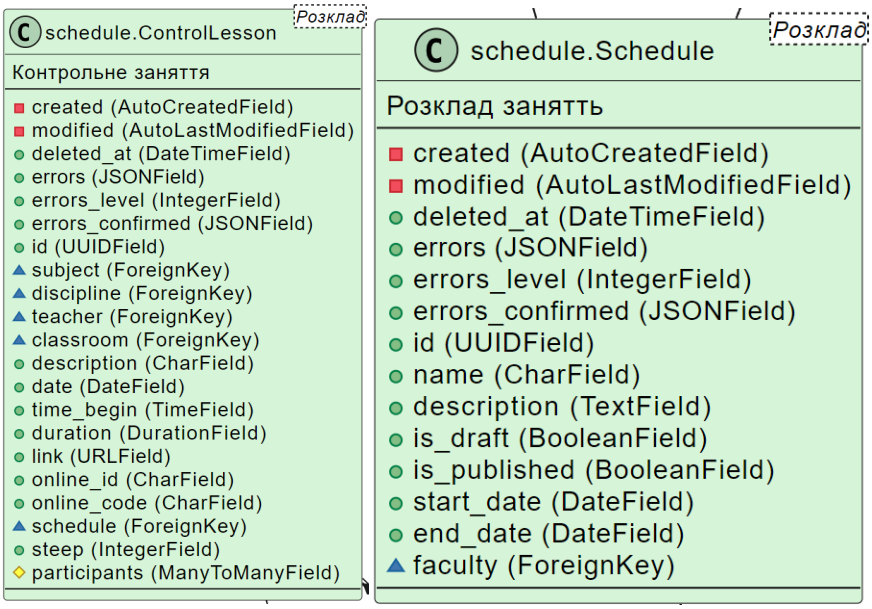


Рисунок 3.13. Структура класів навчального розкладу

Клас «Розклад занять» (англ. Schedule) відповідає за основну інформацію для розкладу на відповідний проміжок часу, який відображений у полях цього класу «дата початку» (англ. start_date), «дата закінчення» (англ. end_date). Розклад може перебувати у двох станах: бути проєктом (англ. is_draft) або бути опублікованим (англ. is_published). Розклад відноситься безпосередньо до факультету.

У свою чергу, за самі заняття відповідають класи «Навчальне заняття» (англ. Lesson) та «Контрольне заняття» (англ. ControlLesson), які мають майже однакову структуру за винятком всього декількох полів (наприклад: «тип» у класі «Навчальне заняття»).

Інші поля цих класів, за винятком суто технічних:

– *teacher* — викладач, що проводить відповідне заняття;

– *subject*, *discipline* — зв’язок із дисциплінами відповідного заняття;

– *link* — покликання на відповідні матеріали у сервісі KSU-online;

– *duration* — тривалість заняття;

– *participants* — безпосередньо учасники цього заняття (здобувачі вищої освіти).

Нарівні із розкладом, навчальними планами, журналами та заліковими відомостями і книжками у системі KSU24 реалізовані класи для підтримки процесів проходження здобувачами вищої освіти навчальної та виробничої практик (рис. 3.14).

Найголовнішою частиною в цьому модулі є клас «Календарний графік проходження практики» (англ. PracticeSchedule), який пов’язаний із «Щоденником практики» студента (англ. PracticeDiary), а також містить інформацію про кількість тижнів відведених на практику, чи було розклад затверджено (поле is_approved), дату коли графік було затверджено (поле approved_on) та ким розклад було затверджено (поле approved_by).

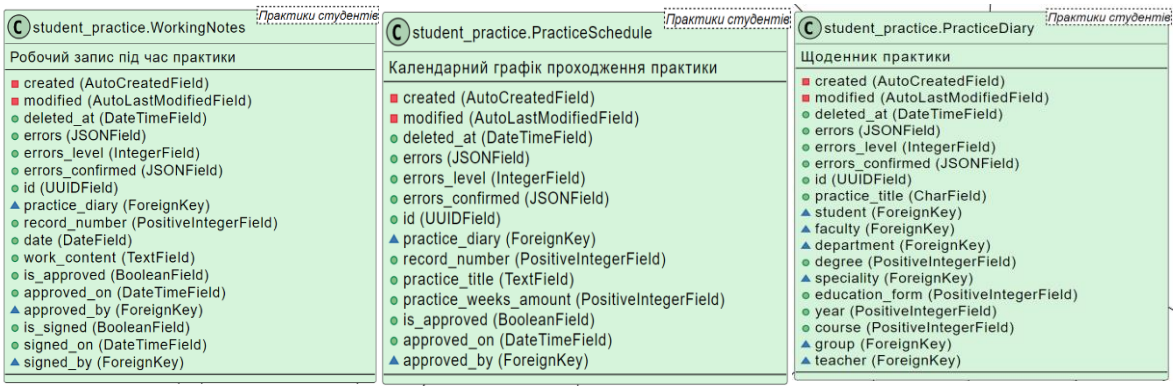


Рисунок 3.14. Класи для журналів та розкладу практики студентів

Клас «Щоденник практики» відповідає за забезпечення зв’язку між календарним планом проходження практики, студентом-практикантом (поле student) та іншими відповідальними відділами (поле department), викладачем керівником (поле teacher), групою практики (group). Також цей клас оперує наступною інформацією:

– курс, на якому проходить практика (поле course);
– назва практики (поле practice_title);
– рік, коли проходить практика (поле year);

До одного «Щоденника практики» відносяться безліч записів класу «Робочий запис під час практики» (англ. WorkingNotes), які відповідають за обробку усієї інформації про хід проходження практики конкретним студентом, хід виконання завдань тощо. Ці робочі записи, як у випадку з відомостями та заліковими книжками, мають певний перелік осіб, які мають їх підтвердити та затвердити. Серед значущих полів цього класу можна виділити поле «Вміст», що зберігає інформацію щодо відміток про виконання конкретної задачі практикантом (англ. Content). Інші поля, чи було конкретне завдання позначено як виконане (поле is_approved), чи взагалі було підписане (поле is_signed), ким було підписано (поле signed_by), коли було підписано (поле signed_on) та дата конкретного запису (поле date).

Дуже важливою складовою частиною сучасного освітнього процесу є зворотній зв’язок між студентами та викладачами. Зворотній зв’язок є визнаним, доказово підтвердженим інструментом підвищення навчальних

результатів. Ефективний зворотній зв'язок, як правило, зосереджений на завданні, предметі та стратегіях самоорганізації: це відгук із чіткою інформацією, що і як покращити. У системі KSU24 комунікація між учасниками навчального процесу реалізована у вигляді системи контролю якості (англ. quality assurance) (рис. 3.15), відгуки студентів про проведені заняття, пройдений курс, а також додано повнофункціональний чат (англ. chat).

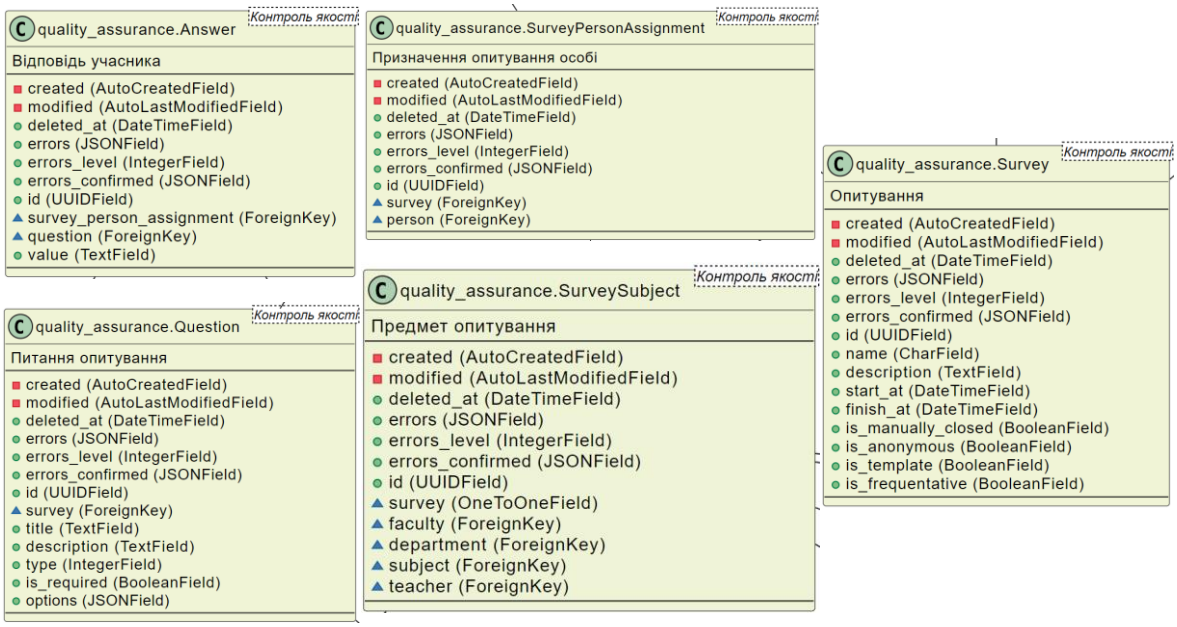


Рисунок 3.15. Структура класів системи контролю якості

Частіше за все, опитування створюються після проведеного заняття і здобувачі вищої освіти мають залишити свої відгуки щодо того, чи були навчальні матеріали для них достатньо зрозумілими, корисними та чи є якісь пропозиції щодо покращення того чи іншого пункту. Ця інформація потім зберігається і обробляється, використовуючи статистичні методи, і надається викладачам, кафедрам, факультетам у вигляді відповідного звіту. Будь-яке опитування є активним певний проміжок часу (за це відповідають поля start_at, finish_at класу «Опитування» (англ. Survey)), після закінчення якого по цьому опитуванню буде створено відповідний звіт. Опитування може бути завершено «вручну» адміністратором або відкрите повторно, якщо існує така необхідність.

Для комунікації між студентами, викладачами, адміністрацією, помічниками з цифровізації тощо. У системі KSU24 використовується модуль чату, структура класів якого наведена далі (рис. 3.16).

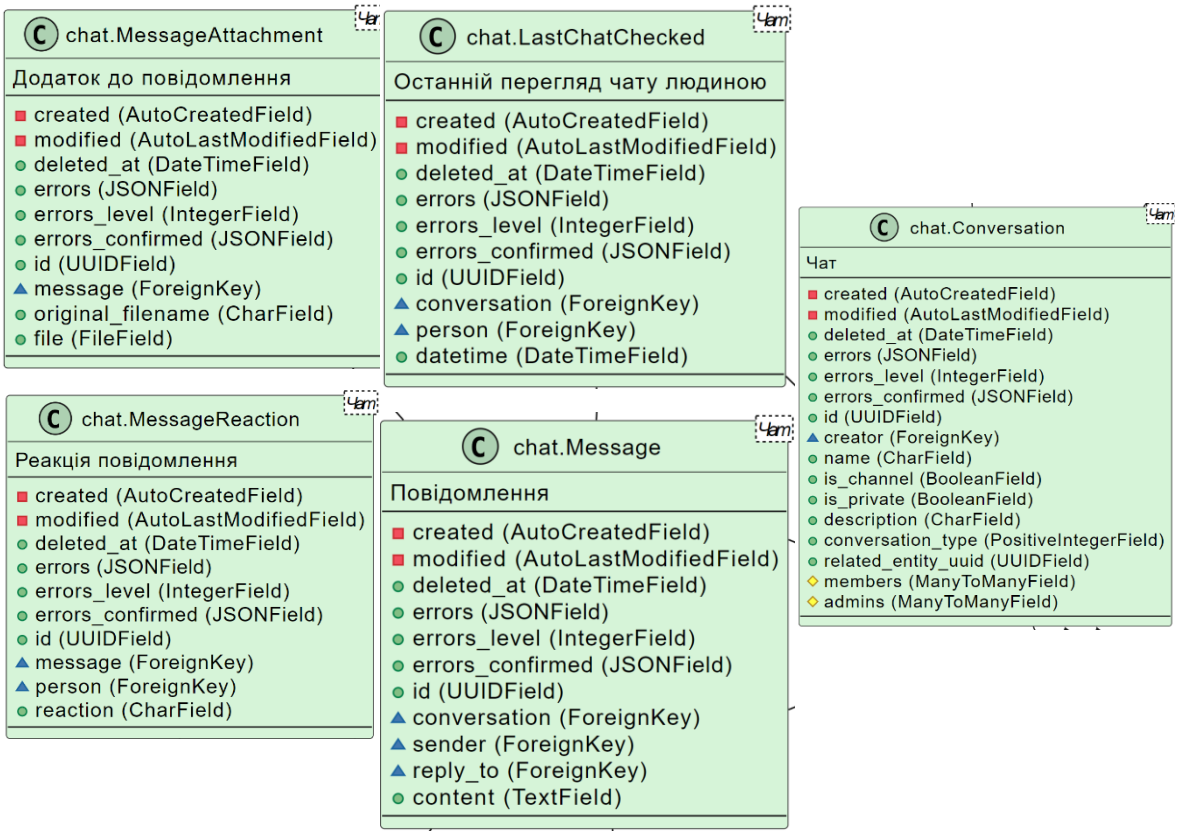


Рисунок 3.16. Структура класів чату системи KSU24

Функціональність модуля чату створена на основі популярних месенджерів. Чат пов'язаний із класом «людина» (human), тому усі користувачі мають можливість ним користуватися. Як і в месенджерах головним є клас «чат» (англ. Conversation), який відповідає за основні налаштування: чи є нова створена розмова приватною (англ. is_private), який вона має тип, хто є учасниками конкретного чату, хто створив чат та ін. Розмова складається із повідомлень, реалізованих відповідним класом «повідомлення» (англ. Message), які можуть мати «вкладення» (англ. Attachment). Також чат зберігає повну історію про те коли, ким було переглянуто повідомлення.

3.3. Клієнтська частина інформаційно-аналітичної системи KSU24

Після того, як для абітурієнта створюється обліковий запис у системі, він отримує відповідну роль. Після цього йому необхідно надати документи для зарахування в якості студента. Документи перевіряються через модуль зв'язку з базою даних ЄДЕБО або, в деяких випадках, можуть бути напряму завантажені з неї. Після того, як усі необхідні документи надійдуть у систему та будуть перевірені, абітурієнт отримує роль студента та доступ до відповідного функціоналу системи, до модулів академічних журналів, розкладу, залікових книжок, опитувань та ін. Документи, які надав студент, він може переглядати у відповідній вкладці у своєму особистому кабінеті. Після завершення процесу реєстрації у системі, користувач отримує обліковий запис та може увійти у систему. Після входу користувач потрапляє на головну сторінку (рис. 3.17).

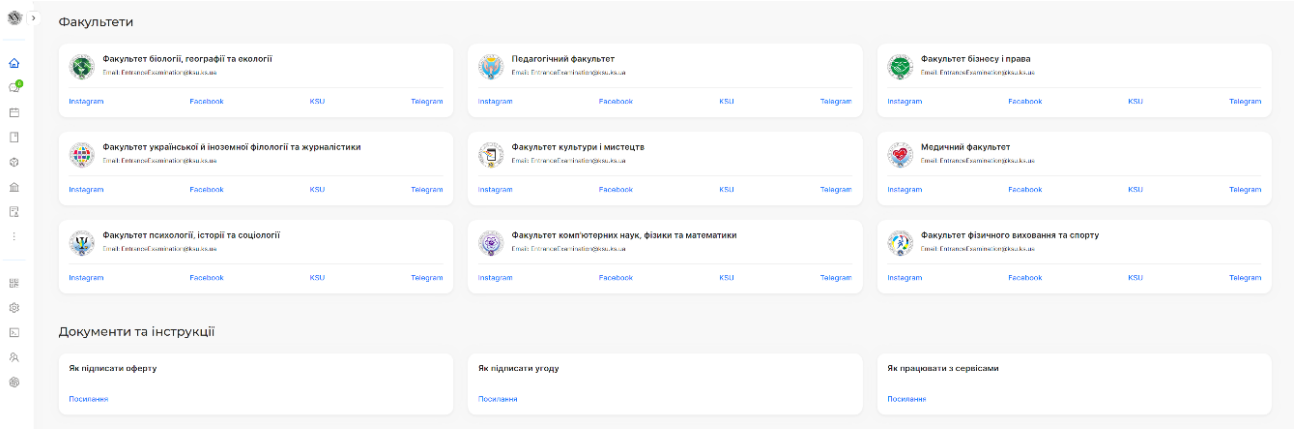


Рисунок 3.17. Головна сторінка системи KSU24,
вигляд для ролі IT помічника.

З цієї сторінки можна потрапити до сторінки необхідного факультету, переглянути потрібні документи та найнеобхідніші інструкції користувача (наприклад, інструкція про те як підписати оферту\угоду та ін). На сторінці кожного факультету, користувачеві доступна основна інформація по факультету, його історія, освітні програми з детальним описом спеціальностей, які здобувачі вищої освіти отримають після закінчення обраного факультета та інформацію про працевлаштування

випускників. Наприклад, наступний вигляд має сторінка факультету комп'ютерних наук, фізики та математики (рис. 3.18).

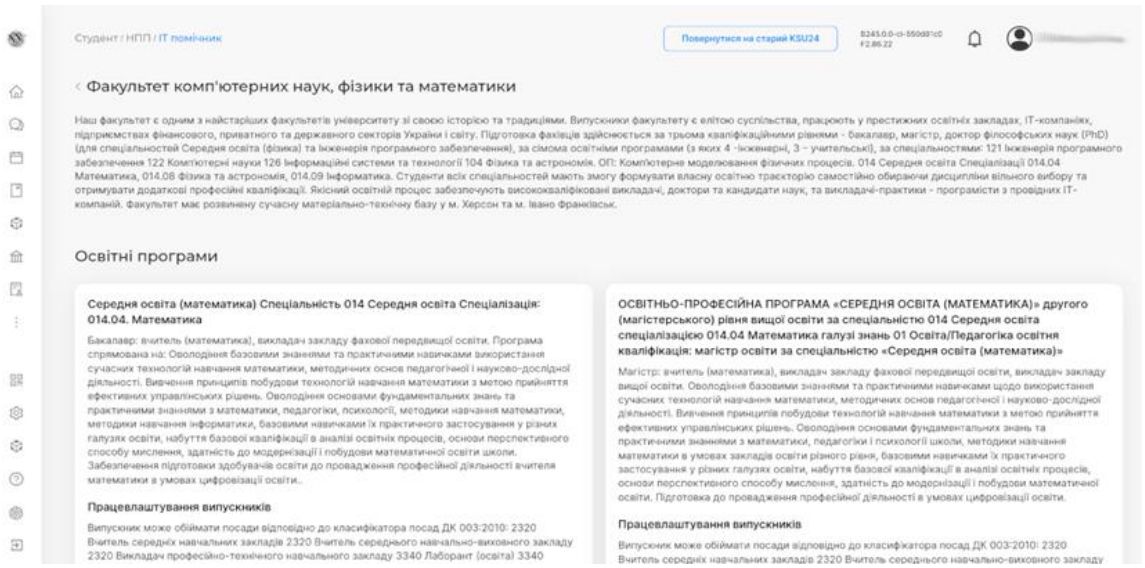


Рисунок 3.18. Вигляд сторінки факультету комп'ютерних наук, фізики та математики

У верхній частині сторінки користувач бачить перелік доступних для нього ролей, між якими можливо під час роботи переключатися за необхідністю.

Також на головній сторінці під блоком «Документів та інструкцій» розташовується блок новин університету (рис. 3.19). Записи у цьому боці фактично дублюють новини з головного сайту Херсонського державного університету і при покликанні на посилання «детальніше» користувач потрапить на сторінку з відповідною новиною на головному сайті.

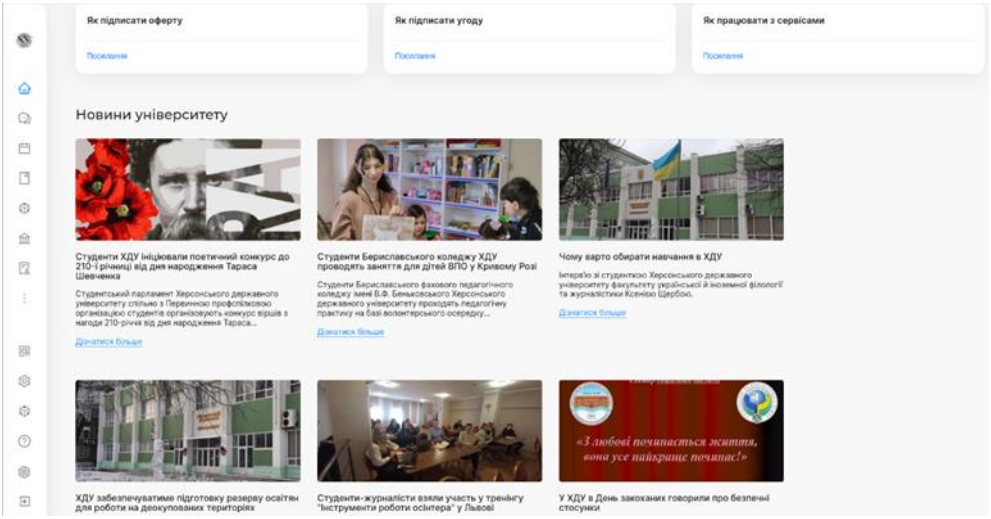


Рисунок 3.19. Новини на сайті системи KSU24

У лівій частині розташоване головне навігаційне меню. Зазвичай воно знаходиться у мінімізованому стані для того щоб не заважати при перегляді основної інформації на сторінці. У розгорнутому вигляді меню має наступний вигляд, презентовано на рис. 3.20.

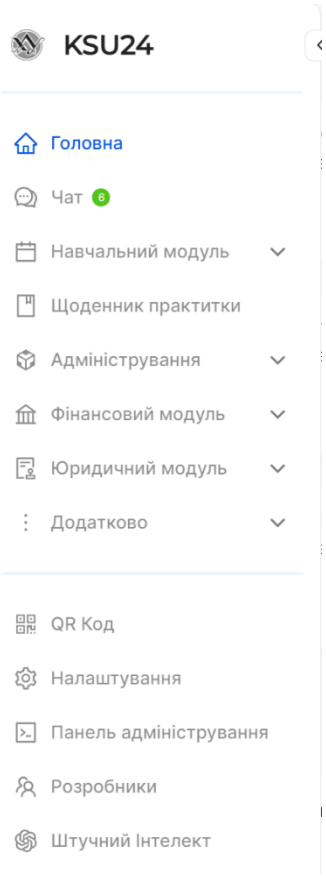


Рисунок 3.20. Головне навігаційне меню системи KSU24

Доступні пункти цього меню можуть відрізнятися залежно від ролі користувача, наприклад, студентові не будуть доступні пункти адміністрування, фінансового та юридичного модулів. Доступ до модуля чату мають усі користувачі і, залежно від їхньої ролі, можуть переглядати та писати повідомлення у відповідні канали (наприклад, здобувачі вищої освіти факультету комп'ютерних наук, фізики та інформатики матимуть доступ до загального чату факультету, до загального чату відповідної кафедри, загального чату їхньої академічної групи). За необхідністю адміністратори можуть створювати нові канали або додавати користувачів у вже існуючи канали. В нижній частині головного меню навігації знаходяться корисні покликання, такі як налаштування, інформація про

команду розробників, qr код для доступу на сайт. Окремо слід виділити пункт «штучний інтелект», який є інтелектуальним помічником заснованим на технології ChatGPT версії 4.0. Це дуже потужний засіб отримання будь-якої корисної інформації, що може знадобитися як студенту (наприклад, у випадку, коли студент під час виконання домашнього завдання стикається із певними труднощами, а викладач у цей час не може надати йому допомоги, то штучний інтелект дозволить отримати необхідну інформацію, і, навіть, допомогти із рішенням, скажімо якщо говорити про факультет комп'ютерних наук, фізики та математики, спеціальностей пов'язаних із програмуванням, то ChatGPT зможе перевірити програмний код, написаний студентом, та надати рекомендацію як виправити можливу помилку), так і викладачу, адміністратору.

Найголовнішим для студента та викладача під час проведення занять є навчальний модуль (рис. 3.21).

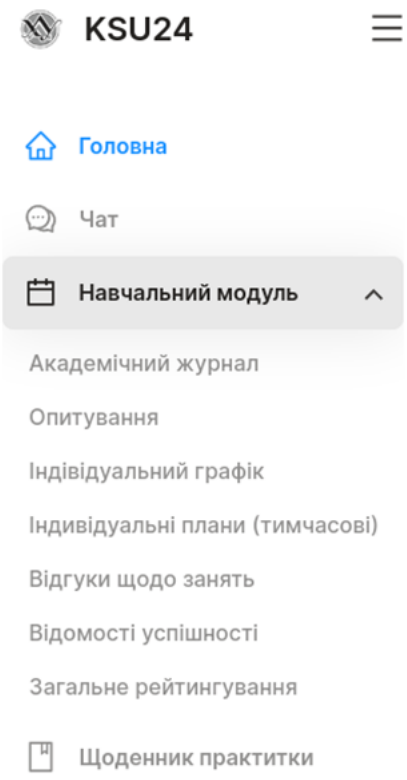


Рисунок 3.21. Пункти меню навчального модуля, для ролі помічника IT

Через це меню користувач може потрапити до усіх необхідних для підтримки навчального процесу модулів. Це, насамперед, розклад занять, «Академічний журнал» та відомості успішності, індивідуальні графіки, а також модулі, що відповідають за зворотній зв'язок: опитування та відгуки щодо занять та модуль загального рейтингу науково-педагогічного працівника у найбільших наукометричних базах даних (Scopus, Google Scholar, Web of Science, Semantic Scholar).

Здобувачі вищої освіти та науково-педагогічні працівники бачать персоналізований розклад занять на відповідній сторінці (рис. 3.22).

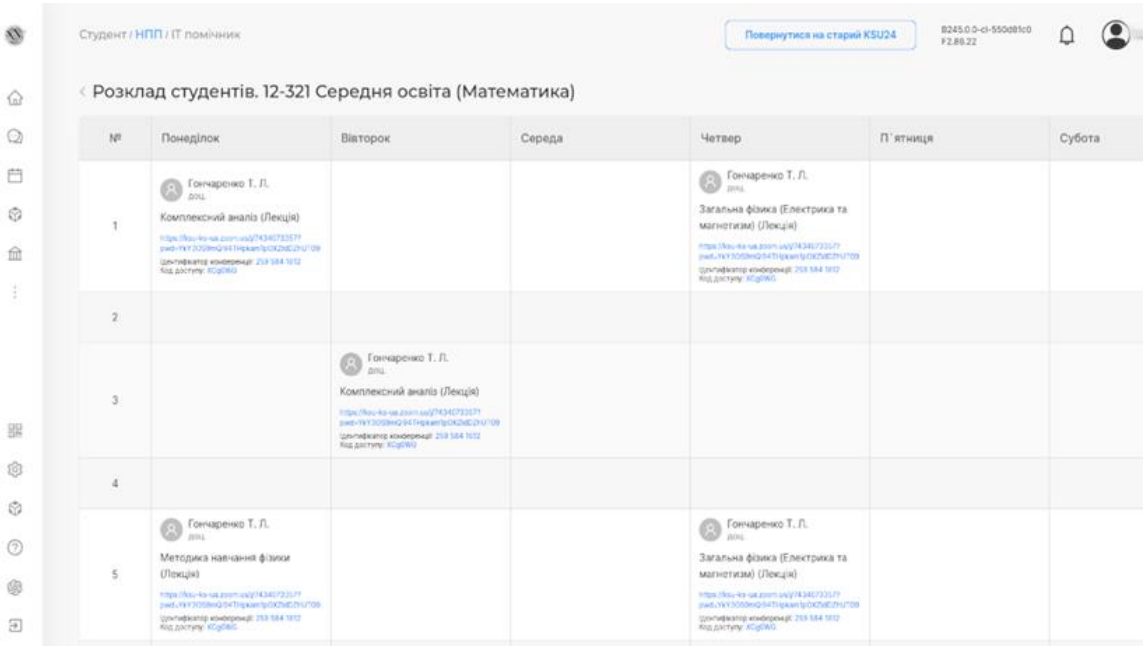


Рисунок 3.22. Загальний вигляд розкладу занять для одного викладача

На сторінці «Академічні журнали» (рис. 3.23) користувач має доступ до усіх журналів відкритих для нього, залежно від ролі у поточний час у вигляді таблиці, може створити новий журнал, натиснувши на відповідну кнопку. За замовченням користувачу будуть показуватись лише активні у поточний час журнали.

У наведеній таблиці відображається загальна інформація про журнали (назва журналу, назва предмету, тип журналу тощо.). Окремо слід відмітити пункт «статус журналу».

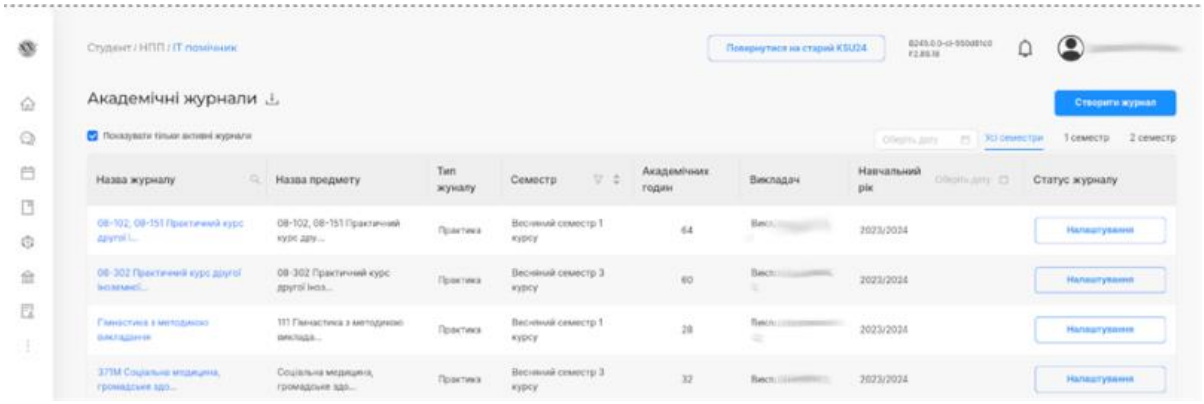


Рисунок 3.23 – Вигляд сторінки академічних журналів

Академічний журнал може бути відкритим, що означає, що заняття по цій дисципліні ще проводяться, або закритим, у цьому випадку викладачу надається можливість сформувати відомість по цьому журналу. Здобувачі вищої освіти мають права лише на перегляд журналів. Викладач може вносити оцінки під час проведення занять. При покликанні на назву обраного журналу, користувач перейде до відповідної сторінки. Для викладача сторінка журналу має наступний вигляд (рис. 3.24).

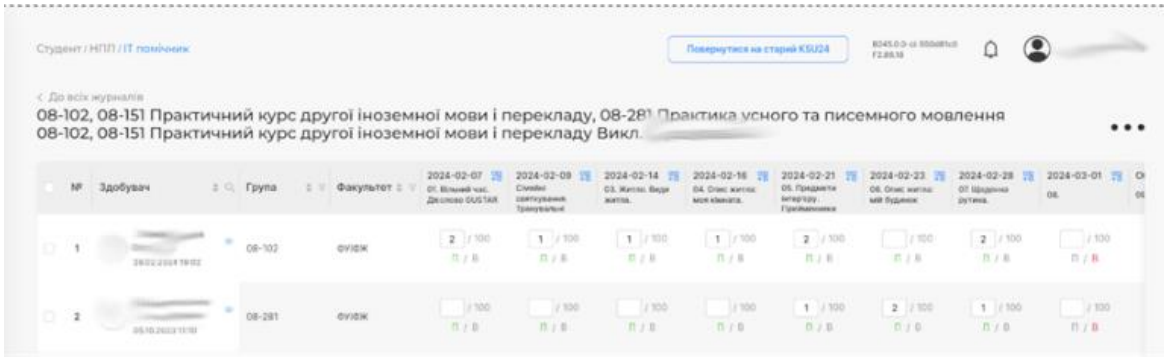


Рисунок 3.24. Вигляд відкритого академічного журналу

У заголовку таблиці журналу поряд із стовпчиками «Здобувач», «Група», «Факультет» розташовані назви тем відповідних занять, дата проведення заняття. Академічний журнал створюється викладачем після покликання на кнопку «Створити журнал». Після цього відкривається відповідне вікно (рис. 3.25), у якому необхідно ввести усю необхідну інформацію: назву журналу, дисципліни, тип, семестр, додавання учасників до журналу.

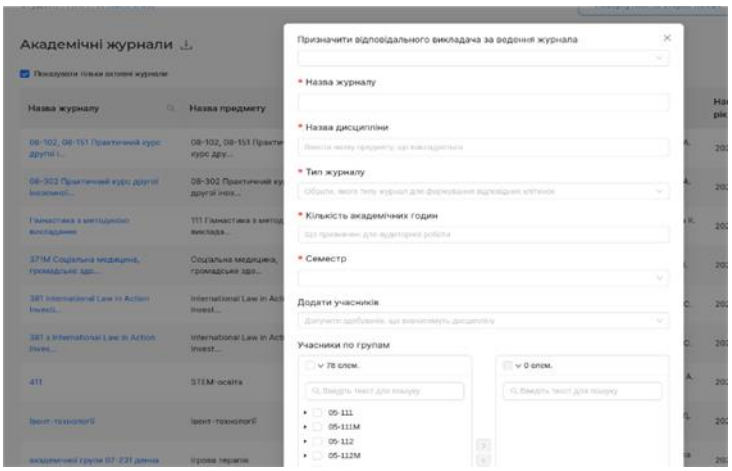


Рисунок 3.25. Вікно створення нового журналу викладачем

Академічний журнал може бути створеним як для одної академічної групи, так і для декількох груп, у випадку якщо предмет одночасно викладається декільком групам одним викладачем (як на прикладі на рис. 3.24). Під час проведення заняття викладач виставляє оцінки у відповідні комірки у 100-бальній системі. В інші системи оцінювання за необхідності система переводить оцінки автоматично.

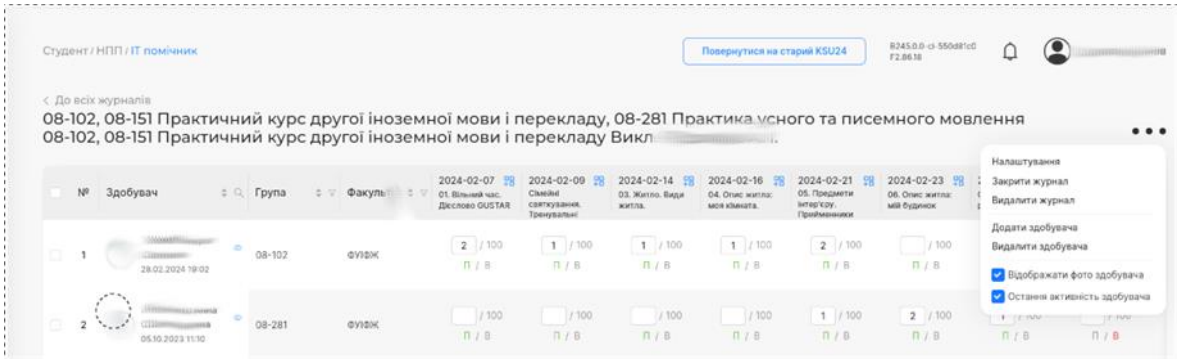


Рисунок 3.26. Меню керування академічним журналом викладача

У правому кутку знаходиться меню налаштування журналу, доступ до якого має лише викладач (рис. 3.26) .Використовуючи це меню, можна налаштовувати як косметичний вигляд (наприклад не відображати фотографії здобувачів), так і керувати журналом. Найголовніші пункти у цьому меню — це покликання для закриття журналу (для можливості формування відомостей) та видалення журналу. Також можливо додавати та видаляти здобувачів в окремих випадках (наприклад, якщо студента було відраховано протягом навчального року, переведено на іншу спеціальність або надано академічну відпустку). У журналі викладач

може подивитись базову інформацію про студента, натиснувши на відповідну іконку поряд з прізвищем студента у таблиці (рис. 3.27). До такої інформації відносяться прізвище, ім’я та по-батькові студента, форма навчання, курс, спеціальність, дати початку та завершення навчання, факультет, спеціальність, форма фінансування, статус освіти в ЄДЕБО, назва кафедри, назва освітньої програми тощо.

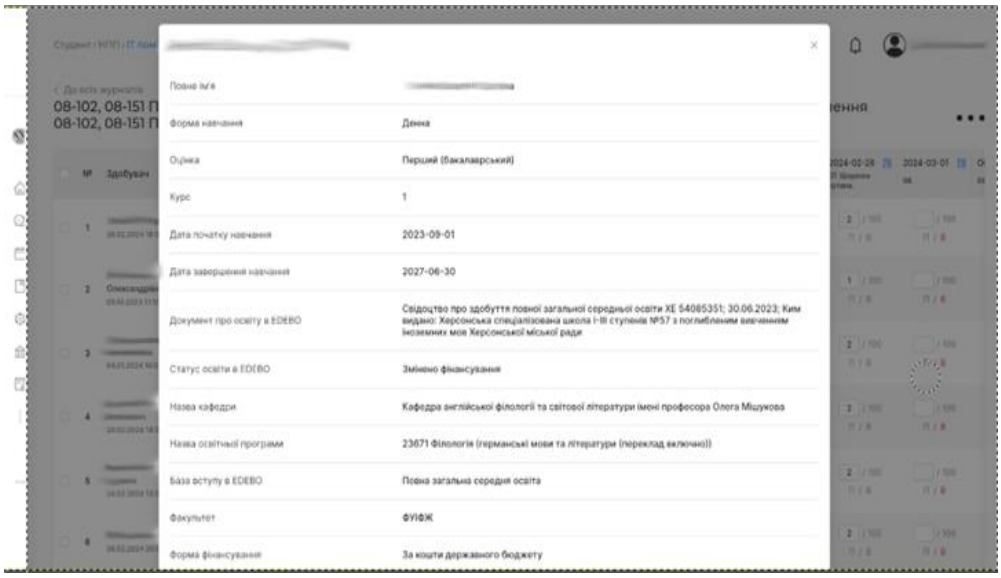


Рисунок 3.27. Загальна інформація про студента, яка доступна для перегляду викладачу з академічного журналу

Академічний журнал також зберігає інформацію про дату останньої активності студента у системі та виводить її під прізвищем студента.

Після проведення усіх запланованих занять та виставлення відповідних оцінок студентам, викладач закриває журнал та отримує можливість створити відомість на основі цього журналу. За необхідності закритий журнал може також бути повторно відкритим для внесення змін. Якщо трапляється подібна ситуація, то після завершення редагування повторно відкритого журналу, викладач знов його закриває та формує нову відомість.

Вже створені відомості доступні до перегляду на відповідній сторінці у вигляді таблиці рис. 3.28.

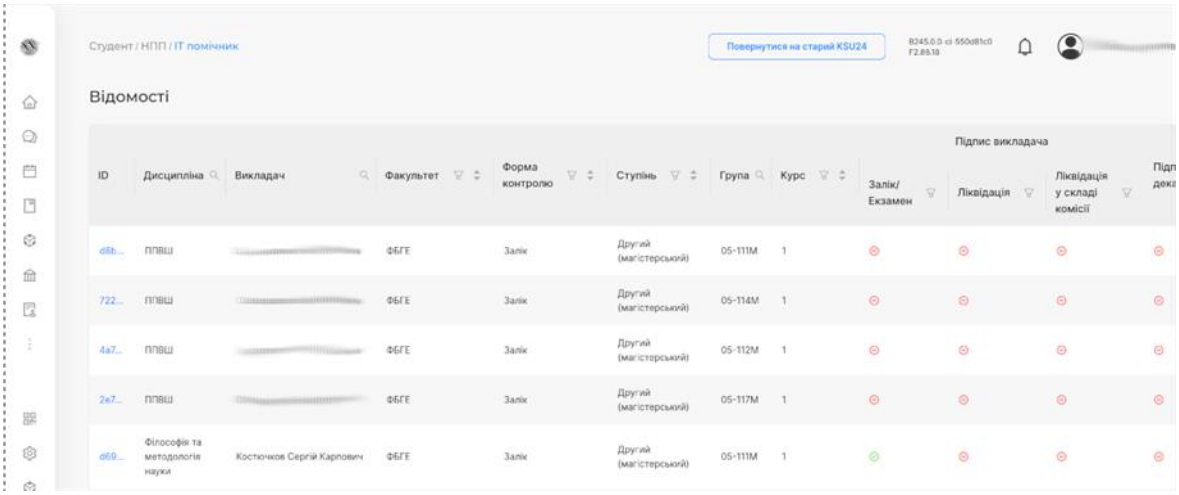


Рисунок 3.28. Сторінка перегляду списку створених відомостей

У таблиці відомостей відображується базова інформація така як: відповідна навчальна дисципліна, хто викладач, форма контролю, група, курс та спеціальна інформація: чи відомість створена для простого заліку або екзамену, для ліквідації або ліквідації у складі комісії. При покликанні на необхідну відомість відкривається відповідна сторінка з усією інформацією, по даній відомості (рис. 3.29).

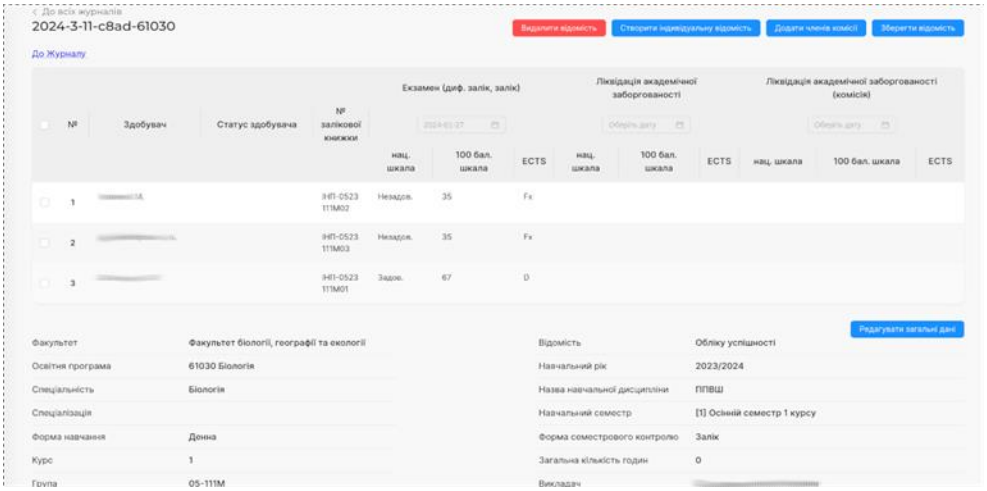


Рисунок 3.29. Сторінка перегляду навчальної відомості

На цій сторінці викладач бачить таблицю зі студентами, номера їх залікових книжок, оцінки, які здобувачі вищої освіти отримали вже у 3-х форматах: за національною шкалою, 100-бальною шкалою та міжнародною шкалою ECTS. При формуванні відомості вона заповнюється оцінками із журналу, на основі якого її було сформовано. З цієї сторінки викладач, який створив відомість має можливість видалити

відомість, якщо на етапі формування було допущено якусь помилку, може редагувати основну інформацію по цій відомості, може додавати та видаляти науково-педагогічних працівників, членів комісії. Після заповнення відомість має бути схвалена або відхилена. За це відповідає навчальний відділ університету. Фінальні та поточні оцінки студентів зберігаються у залікових книжках.

Після закінчення навчального курсу, заняття або взагалі семестру та навчального року у системі KSU24 проводяться відповідні опитування. Загальні результати зворотнього зв'язку від студентів можна побачити на сторінці «Відгуки щодо занять» (рис. 3.30).

На цій сторінці наведені зведені результати по кожному факультету, при покликанні на картку конкретного факультету, відкривається сторінка з інформацією по кожній кафедрі.

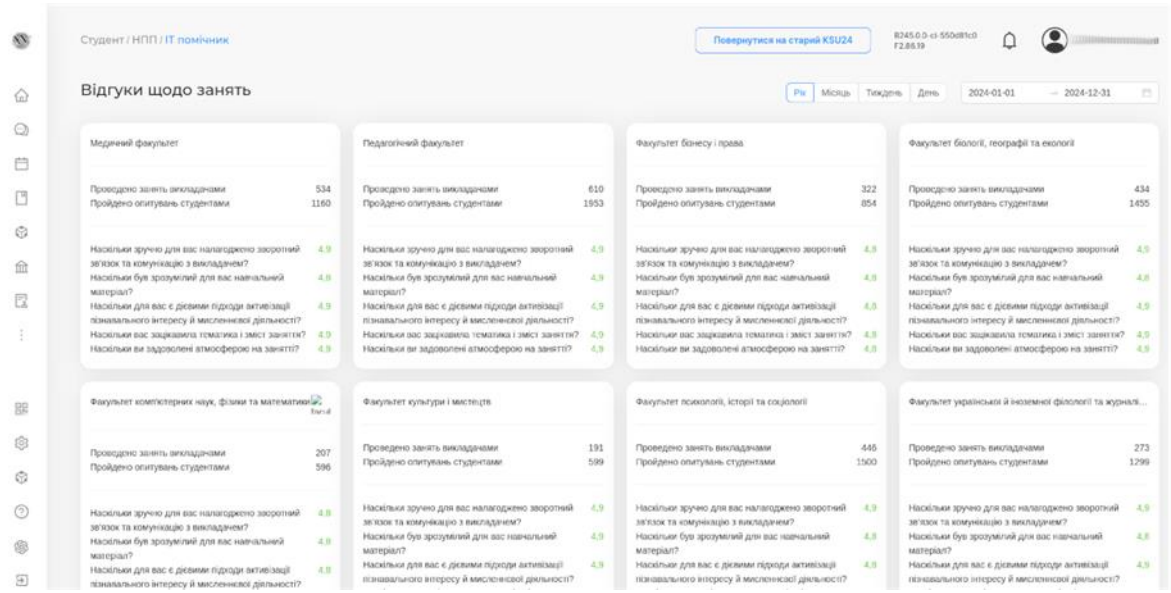


Рисунок 3.30. Сторінка відгуків щодо занять

Модуль «Відгуки щодо занять» надає можливість оцінити задоволеність студентів процесом навчання, виявити слабкі місця на факультетах. Ця узагальнена інформація формується спочатку по кожному викладачу після проведення навчальних занять, заліків екзаменів, виробничої практики тощо. На цій сторінці можна побачити скільки усього занять було проведено викладачами, скільки опитувань було пройдено студентами, який у студентів рівень задоволеності від

занять, наскільки був чи не був зрозумілим навчальний матеріал, якість зворотного зв'язку з викладачами та адміністрацією кафедр, факультетів, закладу освіти в цілому.

Елементи розглянуті у науковому дослідженні [186], було впроваджено у тестовому режимі у модулі "Конференції ZOOM" з аналізом візуальних маркерів (рис 3.31).

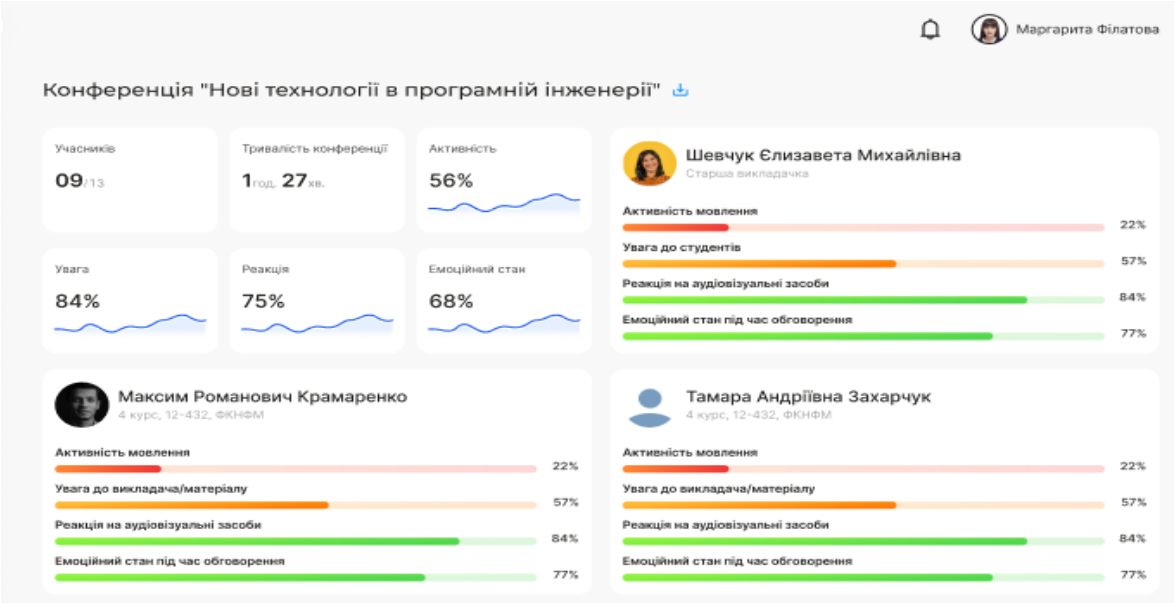


Рисунок 3.31. Модуль аналізу конференцій ZOOM у системі KSU24

Структура класів програмного модуля контроля ZOOM конференцій представлена на рисунку 3.32.

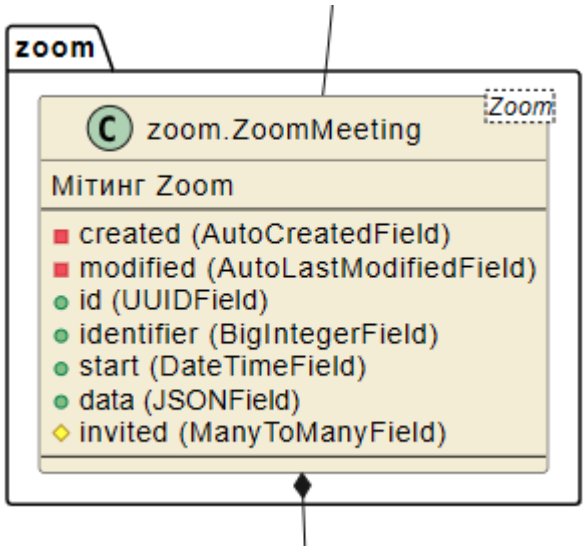


Рисунок 3.32. Структура класу модуля аналізу конференцій ZOOM у системі KSU24

Інтерфейс модуля аналізу конференцій ZOOM представлено на рисунку 3.33

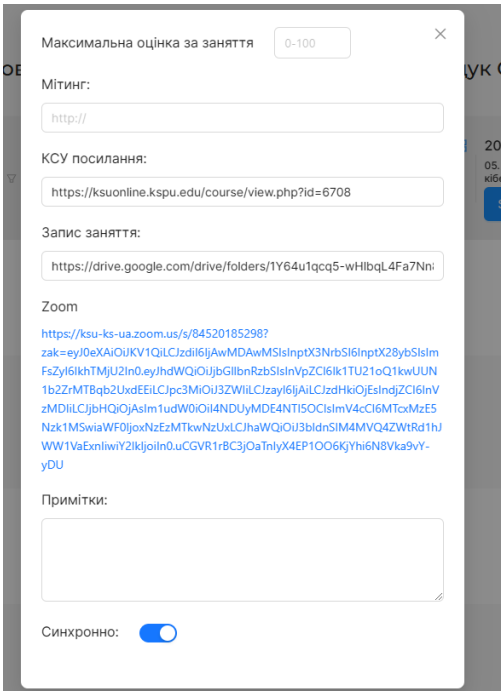


Рисунок 3.33. Інтерфейс користувача модуля аналізу конференцій ZOOM у системі KSU24

До навчального модуля також відносяться індивідуальні плани науково-педагогічних працівників, згруповані по відповідним факультетам, які є головними документами процесів створення планів та звітів науково-педагогічних працівників, відносно різних, окремих видів його діяльності в конкретному навчальному році (рис. 3.34).

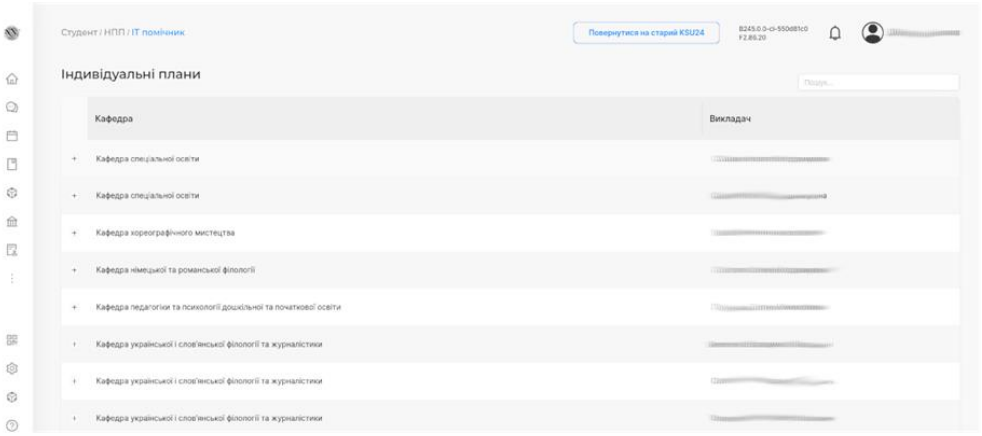


Рисунок 3.34. Сторінка індивідуальних планів науково-педагогічні працівників

Для забезпечення контролю якості освітнього процесу у KSU24 застосовуються напрацювання зі статті [184], що оприлюднює аналіз пошуку найкоротших шляхів з обмеженими ресурсами.

Окрім зазначених модулів, які підтримують безпосередньо основний освітній процес та ведення його документації у KSU24 реалізована система підтримки проходження студентами навчальної та виробничої практик – це модуль «Щоденник практики» (рис. 3.35). У таблиці щоденників практики наведено інформацію про назви щоденників, курс на якому студент проходить практику, кількість тижнів відведених на практику, оцінка за практику та керівник.

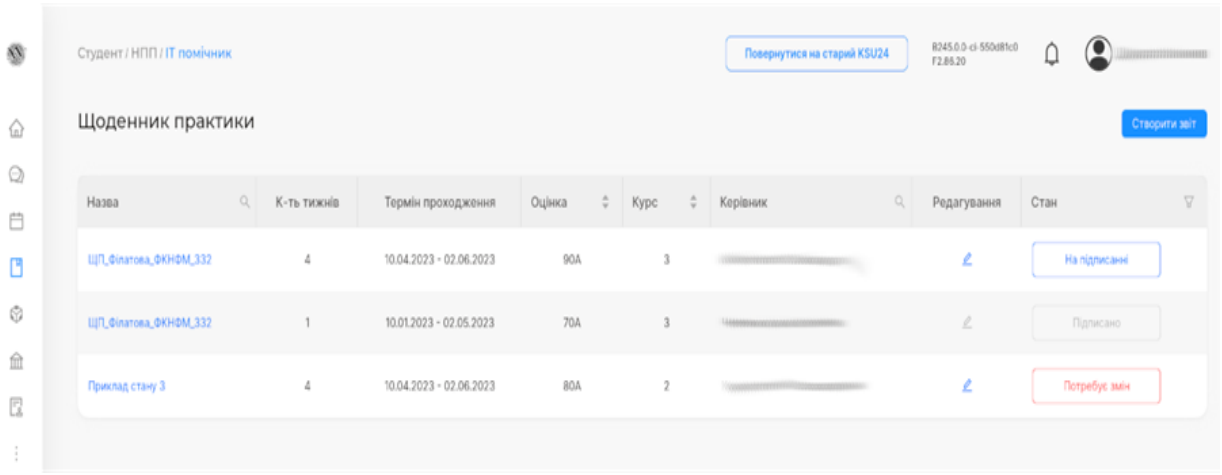


Рисунок 3.35. Загальний вигляд сторінки

модуля «Щоденник практики»

Здобувачі вищої освіти та науково-педагогічні працівники здебільшого користуються саме навчальним модулем. Адміністрація закладу вищої освіти, факультетів та кафедр має доступ до модуля «Адміністрування» (рис. 3.36). Цей модуль дозволяє ведення обліку користувачів, абітурієнтів, оферт, надає доступ до перегляду та редагування інформації про структуру університету, факультетів, кафедр та відділів, перегляд та редагування різноманітних звітів, що створюються за результатами роботи системи KSU24 в цілому та можуть бути згенерованими у будь-якому необхідному вигляді.

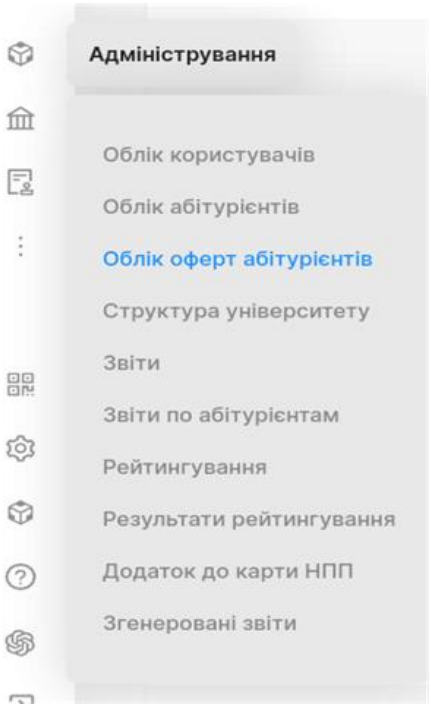


Рисунок 3.36.Меню модуля «Адміністрування» KSU24

На сторінці обліку користувачів (рис. 3.37) адміністратори бачать відповідну технічну інформацію по кожному користувачу, зареєстрованому в системі. Навіть, якщо користувач втрачає доступ до KSU24, його обліковий запис з усією інформацією, що відноситься до цього користувача, зберігається (документи, дипломи, сертифікати, академічні журнали, залікові книжки у випадку якщо користувач мав лише роль студента), і може бути доступним у майбутньому. Адміністратор може за необхідності застосовувати фільтри для того, щоб вивести на сторінку інформацію окремо про користувачів, здобувачів, по окремим академічним групам або працівникам.

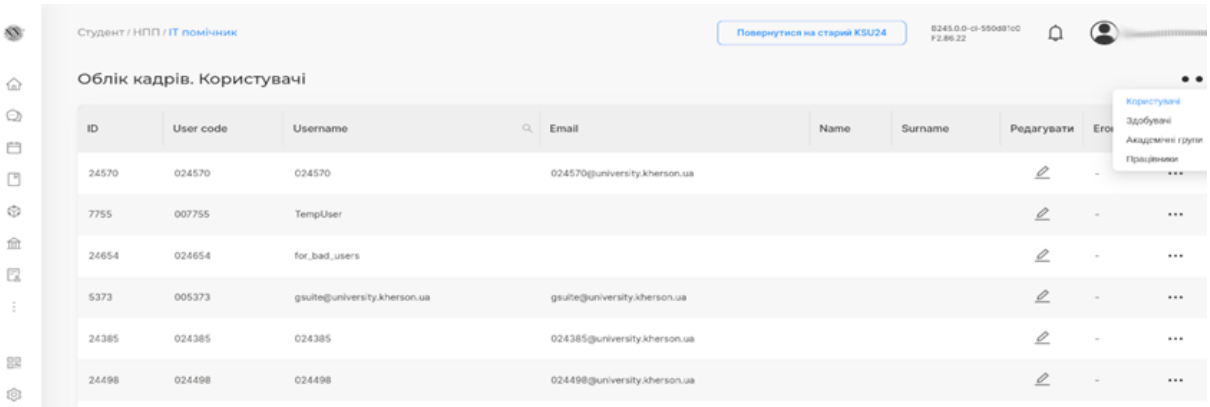


Рисунок 3.37. Сторінка обліку кадрів користувачів

Меню “структура університету” містить основну інформацію по кожному структурному підрозділу (хто є керівником\заступником керівника підрозділу із усією необхідною контактною інформацією), так само по факультетам; перелік у табличному вигляді усіх освітніх програм (рис. 3.38.) та навчальних дисциплін, що викладаються в Херсонському державному університеті.

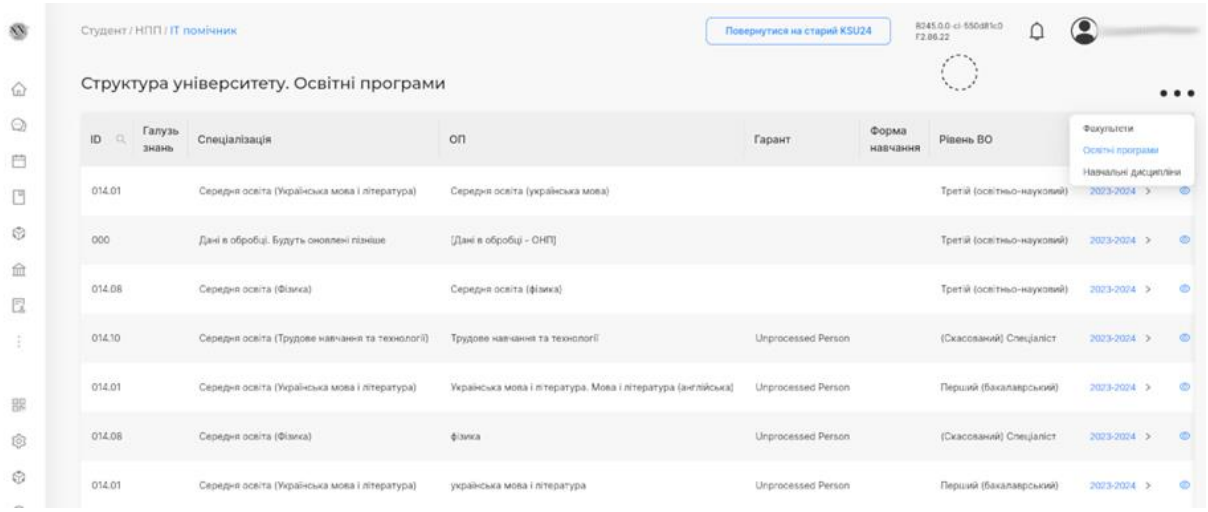


Рисунок 3.38. Освітні програми в структурі університету

Особливо, слід зазначити, важливу функцію уособлення (англ. impersonate), якою можуть користуватися лише адміністратори системи. Уособлення користувача означає здатність системи тимчасово надавати доступ іншому обліковому запису користувача, як правило, адміністратору або персоналу служби підтримки. Ця функціональність є надзвичайно корисною для усунення несправностей, тестування або вирішення проблем користувачів без необхідності введення облікових даних початкового користувача. Таким чином, адміністратор має можливість зайти у систему від імені будь-якого користувача для того, щоб якнайшвидше розібратись у складній ситуації та надати необхідну допомогу, у випадку виникнення в останнього складнощів у користуванні системою.

Окремо також адміністрація має доступ до детальної інформації саме про абітурієнтів: їх контактні дані, подані документи, обрані спеціальність та спеціалізація, публічні оферти та статус їх підписання, як

окремо за кожним пунктом, так і у вигляді узагальненого звіту (рис. 3.39).

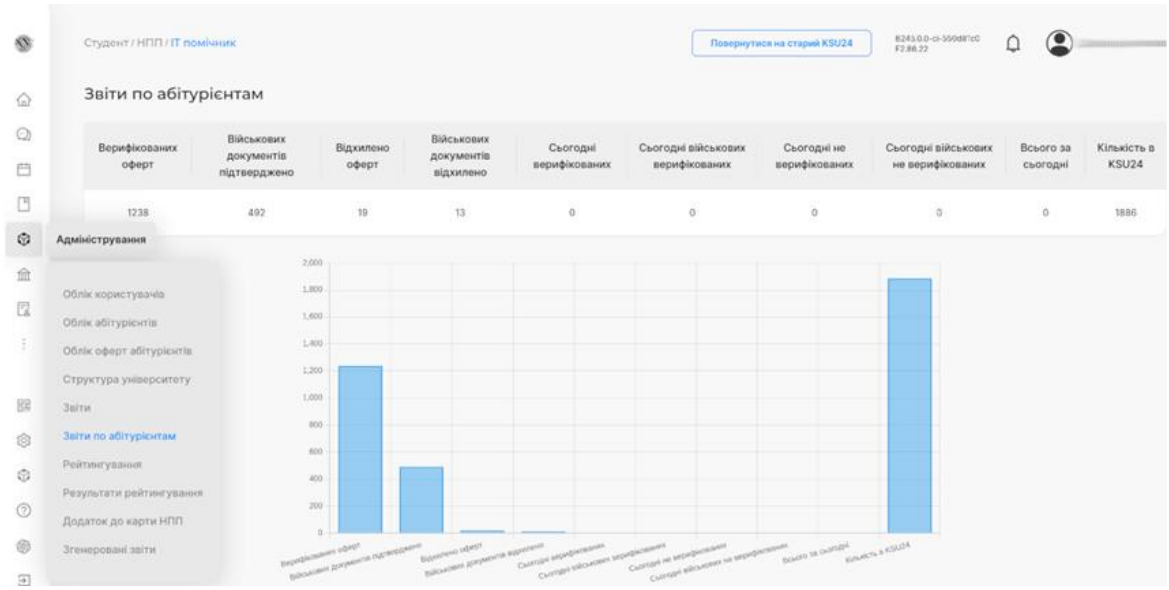


Рисунок 3.39. Сторінка звітів по абітурієнтам

З меню адміністрування також є можливість переглядати рейтингування, як загалом по факультетам, так і по кожному науково-педагогічному працівнику. Рейтингування НПП здійснюється за наступними показниками:

1. «Наявність не менше п’яти публікацій у періодичних наукових виданнях, що включені до переліку фахових видань України, до наукометричних баз, зокрема Scopus, Web of Science Core Collection» [187].

2. «Наявність одного патенту на винахід або п’яти деклараційних патентів на винахід чи корисну модель, включаючи секретні, або наявність не менше п’яти свідоцтв про реєстрацію авторського права на твір» [187].

3. «Наявність виданого підручника чи навчального посібника (включаючи електронні) або монографії (загальним обсягом не менше 5 авторських аркушів), в тому числі видані у співавторстві (обсягом не менше 1,5 авторського аркуша на кожного співавтора)» [187].

4. «Наявність виданих навчально-методичних посібників/посібників для самостійної роботи здобувачів вищої освіти та дистанційного навчання, електронних курсів на освітніх платформах

ліцензіатів, конспектів лекцій / практикумів / методичних вказівок/рекомендацій/ робочих програм, інших друкованих навчально-методичних праць загальною кількістю три найменування» [187].

5. «Захист дисертації на здобуття наукового ступеня» [187].

6. «Наукове керівництво (консультування) здобувача, який одержав документ про присудження наукового ступеня» [187].

7. «Участь в атестації наукових кадрів як офіційного опонента або члена постійної спеціалізованої вченої ради, або члена не менше трьох разових спеціалізованих вчених рад» [187].

8. «Виконання функцій (повноважень, обов'язків) наукового керівника або відповідального виконавця наукової теми (проєкту), або головного редактора/члена редакційної колегії/експерта (рецензента) наукового видання, включеного до переліку фахових видань України, або іноземного наукового видання, що індексується в бібліографічних базах» [187].

9. «Робота у складі експертної ради з питань проведення експертизи дисертацій МОН або у складі галузевої експертної ради як експерта Національного агентства із забезпечення якості вищої освіти, або у складі Акредитаційної комісії, або міжгалузевої експертної ради з вищої освіти Акредитаційної комісії, або трьох експертних комісій МОН / зазначеного Агентства, або Науково-методичної ради / науково-методичних комісій (підкомісій) з вищої або фахової передвищої освіти МОН, наукових / науково-методичних / експертних рад органів державної влади та органів місцевого самоврядування, або у складі комісій Державної служби якості освіти із здійснення планових (позапланових) заходів державного нагляду (контролю)» [187].

10. «Участь у міжнародних наукових та/або освітніх проектах, залучення до міжнародної експертизи, наявність звання «суддя міжнародної категорії» [187].

11. «Наукове консультування підприємств, установ, організацій не менше трьох років, що здійснювалося на підставі договору із закладом вищої освіти (науковою установою)» [187].

12. «Наявність апробаційних та/або науково-популярних, та/або консультаційних (дорадчих), та/або науково-експертних публікацій з наукової або професійної тематики загальною кількістю не менше п’яти публікацій» [187].

13. «Проведення навчальних занять із спеціальних дисциплін іноземною мовою (крім дисциплін мовної підготовки) в обсязі не менше 50 аудиторних годин на навчальний рік» [187].

14. «Керівництво студентом, який зайняв призове місце на І або ІІ етапі Всеукраїнської студентської олімпіади (Всеукраїнського конкурсу студентських наукових робіт), або робота у складі організаційного комітету/журі Всеукраїнської студентської олімпіади (Всеукраїнського конкурсу студентських наукових робіт), або керівництво постійно діючим студентським науковим гуртком/проблемною групою; керівництво студентом, який став призером або лауреатом Міжнародних, Всеукраїнських мистецьких конкурсів, фестивалів та проектів, робота у складі організаційного комітету або у складі журі міжнародних, всеукраїнських мистецьких конкурсів, інших культурно-мистецьких проектів (для забезпечення провадження освітньої діяльності на третьому (освітньо-творчому) рівні); керівництво здобувачем, який став призером або лауреатом міжнародних мистецьких конкурсів, фестивалів, віднесених до Європейської або Всесвітньої (Світової) асоціації мистецьких конкурсів, фестивалів, робота у складі організаційного комітету або у складі журі зазначених мистецьких конкурсів, фестивалів); керівництво студентом, який брав участь в Олімпійських, Паралімпійських іграх, Всесвітній та Всеукраїнській Універсіаді, чемпіонаті світу, Європи, Європейських іграх, етапах Кубка світу та Європи, чемпіонаті України; виконання обов’язків тренера, помічника

тренера національної збірної команди України з видів спорту; виконання обов'язків головного секретаря, головного судді, судді міжнародних та всеукраїнських змагань; керівництво спортивною делегацією; робота у складі організаційного комітету, суддівського корпусу» [187].

15. «Керівництво школярем, який зайняв призове місце III-IV етапу Всеукраїнських учнівських олімпіад з базових навчальних предметів, II-III етапу Всеукраїнських конкурсів-захистів науково-дослідницьких робіт учнів - членів Національного центру «Мала академія наук України»; участь у журі III-IV етапу Всеукраїнських учнівських олімпіад з базових навчальних предметів чи II-III етапу Всеукраїнських конкурсів-захистів науково-дослідницьких робіт учнів - членів Національного центру «Мала академія наук України» (крім третього (освітньо-наукового/освітньо-творчого) рівня)» [187].

16. «Наявність статусу учасника бойових дій (для вищих військових навчальних закладів, закладів вищої освіти із специфічними умовами навчання, військових навчальних підрозділів закладів вищої освіти)» [187].

17. «Участь у міжнародних операціях з підтримання миру і безпеки під егідою Організації Об'єднаних Націй (для вищих військових навчальних закладів, закладів вищої освіти із специфічними умовами навчання, військових навчальних підрозділів закладів вищої освіти)» [187].

18. «Участь у міжнародних військових навчаннях (тренуваннях) за участю збройних сил країн - членів НАТО (для вищих військових навчальних закладів, військових навчальних підрозділів закладів вищої освіти)» [187].

19. «Діяльність за спеціальністю у формі участі у професійних та/або громадських об'єднаннях» [187].

20. «Досвід практичної роботи за спеціальністю не менше п’яти років (крім педагогічної, науково-педагогічної, наукової діяльності)» [187].

21. «Відомості про підвищення кваліфікації (власне професійне та міжнародне протягом останніх 5-ти років, затверджених наказом по університету): найменування закладу, тема, вид документа, кількість кредитів, номер наказу по ХДУ» [187].

22. «Стаж науково-педагогічної роботи (повних років)» [187].

Цей список розміщено на відповідній сторінці проекту KSU24. Він регулярно оновлюється та доповнюється.

3.4. Ефективна генерація звітної інформації в інформаційно-аналітичній системі KSU24

Однією із головних задач працівників адміністративних відділів, наприклад, загального відділу, відділу забезпечення якості освіти — це створення та передача відповідним відділам, керівництву закладу вищої освіти, відповідальним працівникам Міністерства освіти та науки України, різноманітної звітної інформації (рис 3.40).

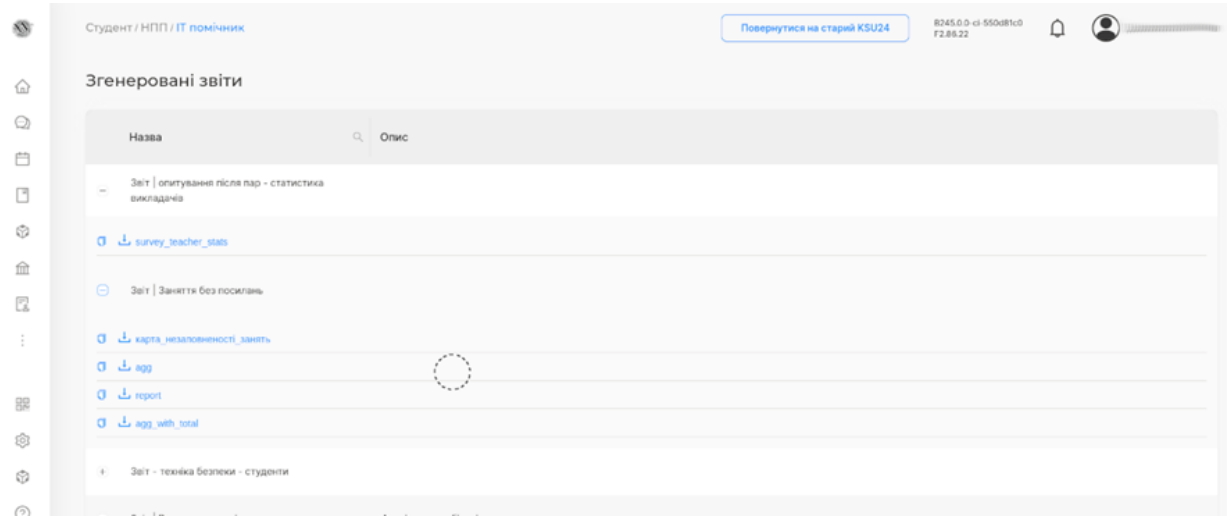


Рисунок 3.40. Згенеровані звіти у системі KSU24

У системі KSU24 для створення відповідних звітів за необхідними параметрами використовується спеціалізована бібліотека «Python Pandas» [181].

«Pandas» (у стилі pandas) — це бібліотека програмного забезпечення, написана на мові програмування Python для обробки та аналізу даних, яка надає структури даних і операції для роботи з числовими таблицями та часовими рядами.

Система «Pandas» побудована навколо структур даних, які мають назву «Series and DataFrames». Інформація для цих колекцій імпортується з різних форматів файлів, таких як значення, розділені комами, JSON, Parquet, таблиці або запити бази даних SQL і Microsoft Excel [182].

Структура «Series» — це одновимірна структура даних, побудована на основі масиву, кожна точка даних у якому має відповідну мітку. Набір цих міток називається індексом. Рядок може бути використаним у звичайній арифметичній операції, наприклад, як у операторі «series_3 = series_1 + series_2»: цей вираз відсортує точки даних із відповідними значеннями індексу в «series_1 і series_2», після того додасть їх один до одного для створення нових значеннь в «series_3». «DataFrame» — це двовимірна структура даних із рядків і стовпців, схожа на електронну таблицю та аналогічна словнику у мові Python, що зіставляє імена стовпців (ключі) із «Series» (значення) з кожним «Series», що мають спільний індекс.

Користувачі мають можливість перетворювати або підсумовувати інформацію, використовуючи довільні функції. Оскільки «Pandas» побудовано на основі бібліотеки «NumPy», усі функції з цієї бібліотеки також працюють із цими структурами даних. «Pandas» також містить вбудовані операції для арифметики, маніпуляції з рядками та підсумкові статистичні дані, такі як середнє значення, медіана та стандартне відхилення. Ці вбудовані функції призначені для обробки відсутніх даних, зазвичай, представлених значенням із плаваючою комою. Підмножини даних можна вибирати за назвою стовпця, індексом або логічними виразами.

До складу «Pandas» включено підтримку часових рядів, наприклад

можливість інтерполяції значень та фільтрації за допомогою діапазону часових позначок (наприклад, «data['1/1/2023':'2/2/2023']» поверне усі дати у проміжку від 1 січня до 2 лютого).

За замовчуванням індекс в «Pandas» — це ряд цілих чисел, що починаються з 0, подібно до індексів масивів у мові програмування Python. Однак у якості індексів можуть бути використані будь-які типи даних із бібліотеки «NumPy», включаючи числа з плаваючою точкою, мітки часу або рядки.

Синтаксис Pandas для зіставлення значень індексу з релевантними даними є таким самим, який використовується у мові Python для зіставлення ключів словників зі значеннями. На відміну від ключів словника унікальність значень індексів не гарантується. Якщо «Series» використовує значення відповідного індексу «a» для кількох точок даних, тоді s['a'] поверне нову серію, що містить усі відповідні значення. Імена стовпців структури “DataFrame” зберігаються так само як і індекси.

У «Pandas» підтримуються ієрархічні індекси з кількома значеннями для однієї точки даних. Індекс із такою структурою, який має назву «мульти-індекс» дозволяє для однієї структури «DataFrame» представляти кілька вимірів, подібно до зведеної таблиці в Microsoft Excel.

Розроблений компонент «Редактор коду (Shell)»:

- дозволяє швидко імплементовувати одноразові звіти;
- гнучко імпортувати і експортувати інформацию з системи;
- формувати візуалізації різних видів;
- надаваті користувачам доступ до використання підготовлених звітів;
- виконувати одноразові адміністративні дії (швидкі одноразові звіти, створення автоматичних сповіщень, опитувань, імпорт-експрорт даних) (Додаток Е).

Перелік звітів:

- Звіт | Місцеположення
- Звіт | Заняття без посилань
- Звіт | опитування по журналам
- Звіт | фото студентів
- Звіт | Техніка безпеки
- Звіт | результати сесії
- Звіт | Журнал реєстрації відомостей
- Звіт | Авторський договір
- Звіт | Кодекс академічної доброчесності
- Звіт | Курси KSU Online
- Звіт | результати опитувань
- Звіт | накопичуваність балів та відвідуваність
- Звіт | Відсоток накопичувальності оцінок «додаток В»
- Звіт | присутність студентів
- Звіт | Співробітники
- Звіт | Вступна кампанія «додаток Б»
- Звіт | БД «додаток Г»

Застосування технології Pandas дозволяє ефективне та оптимальне створення звітів за будь-якими обраними показниками, використовуючи досвід описаний у статті [185].

Висновки до 3 розділу

1. Головним архітектурним шаблоном для розробки серверної частини віртуального освітнього середовища — є модель-вид-представлення, було обрано каркас інтернет застосунків — Django, мови програмування Python.

2. Відповідно до архітектурного шаблону модель-вид-представлення та особливостей його реалізації у каркасі інтернет застосунків Django, було розроблено відповідну структуру класів для

реалізації усіх необхідних функціональних можливостей серверної частини віртуального освітнього середовища KSU24.

3. Для розробки клієнтської частини інтернет-застосунка віртуального освітнього середовища використано каркас інтернет застосунків, побудови користувацьких інтерфейсів ReactJS та розроблено відповідні користувацькі інтерфейси.

4. У системі «KSU24» використовуються найсучасниші інструменти та бібліотеки мови програмування “Python” для реалізації максимально гнучкої та потужної системи генерації звітної інформації.

ВИСНОВКИ

Результатами дисертаційного дослідження на тему «Розробка системи управління процесам якості освіти університету за допомогою сервісів» стали: повний спектр функціональних та системних вимог до побудови віртуальних освітніх середовищ для закладів вищої освіти України та світу, та готова програмна платформа, що інтегрована у бізнес-процеси Херсонського державного університету, знаходиться в активному оновлені та доповнюється новими функціональними можливостям, а також готується до розширення на інфраструктуру інших закладів вищої освіти України та світу. В ході дисертаційного дослідження було розв'язано наступні задачі:

- проаналізовано сучасні підходи та інформаційні технології підтримки навчального процесу у закладах вищої освіти, в результаті чого було сформульовано функціональні вимоги для побудови нової системи;
- охарактеризовано сучасні підходи та інформаційні технології для підтримки економічно-правових процесів у закладах вищої освіти, в результаті чого було реалізовано відповідні програмні модулі системи «KSU24», додано ролі користувачів для співробітників юридичного відділу та бухгалтерії, та розроблено і впроваджено модулі інтеграції із відповідними сервісами, що використовуються в Україні;
- застосовано, проаналізовано та впроваджено методології побудови віртуальних навчальних середовищ, на основі яких будуються сучасні інформаційні середовища у закладах вищої освіти України;
- створено технічне завдання, тестовий план, надано обґрунтування вибору програмних технологій та архітектурних рішень для розробки та впровадження віртуального навчального середовища у заклади вищої освіти України. Описано повний досвід Херсонського державного університету із розробки та впровадження віртуальних освітніх середовищ у бізнес-процеси закладів вищої освіти, для

можливості подальшого масштабування системи на інші заклади вищої освіти;

- спроектовано та розроблено серверну частину інтернет застосунка та структуру бази даних програмного засобу для підтримки навчальних, адміністративних та інших бізнес процесів закладу вищої освіти;

- спроектовано та розроблено інтерфейси, та клієнтську частину програмного засобу відповідно до сучасної методології створення користувацьких інтерфейсів – «Ant-design».

В *першому розділі* розглянуто й проаналізовано головні складові частини сучасних віртуальних навчальних середовищ. Описані сучасні методології та підходи, які використовуються при побудові віртуальних навчальних середовищ в цілому і студентських навчальних середовищ. Розглянуто системи керування навчанням та механізми їх інтеграції із іншими програмними платформами, для керування та підтримки безпосередньо самих освітніх курсів, та забезпечення для усіх учасників навчального процесу доступу до усіх типів навчальних матеріалів: відео, курсів, семінарів, документів. У якості основи для програмного комплексу який об’єднує інформаційні технології, методологічні та практичні аспекти керування та підтримки навчального процесу, а також різноманітні аспекти бізнес процесів сучасного вищого навчального закладу, вперше було запропоновано використовувати досвід із розробки «систем керування взаємовідносинами клієнтів». Наведені варіанти застосування методів і технологій інтелектуального аналізу даних в віртуальних навчальних середовищах. Новими є теоретичні та технологічні підходи для інтеграції основних типів моделей інтелектуальних навчальних систем до віртуального навчального середовища університету для забезпечення надання негайних відповідей та інструкцій студентам у тих випадках коли негайне втручання викладача-людини неможливе у поточний проміжок часу.

У *другому розділі* наведено методології і технології проектування, розробки, контролю якості та впровадження віртуальних навчальних середовищ у вищі навчальні заклади України. Розглядаються архітектурні принципи та технологічні рішення побудови сучасної інформаційно-аналітичної системи управління закладами вищої освіти. У якості основних технологій розробки зі сторони сервера обрано:

- мову програмування «Python»;
- фреймворк Django, який реалізовує актуальну версію шаблону проектування «модель-вид-контролер»;
- базу даних PostgreSQL.

Для реалізації клієнтської частини використовуються: мова програмування «JavaScript», сучасний фреймворк “React”, із застосуванням методології побудови користувацьких інтерфейсів «Ant-design», та технологія «NodeJS», яка дозволяє значно оптимізувати та пришвидшити роботу користувачів системи завдяки використанню технології обробки графіки на стороні сервера.

Наведено методології та технології контролю якості, що використовуються під час розробки та вдосконалення віртуального навчального середовища «KSU24», як з боку команди розробників під час впровадження нових функціональних можливостей до системи, так і зі сторони тестувальників та кінцевих користувачів системи, які надають зворотній зв’язок, щодо коректної реалізації та роботи відповідних програмних модулей.

Новою є пропозиція використання технологій блокчейн, яка може надати платформам електронного навчання можливість отримання засобів надання захищених від підробки записів про досягнення студентів, їх сертифікатів, кваліфікацій, та в майбутньому, дипломів про вищу освіту.

У *третьому розділі* описуються основні програмні модулі та їх графічні інтерфейси із якими безпосередньо працюють усі користувачі

(здобувачі вищої освіти, науково-педагогічні працівники, адміністрація) в залежності від відповідних ролей у системі, а саме:

- особистий кабінет користувача, доступ до відповідних функцій у якому відрізняється в залежності від ролі: студент, викладач, Іт-помічник, адміністратор тощо.;
- навчальний модуль, до якого відносяться академічні журнали груп, навчальні відомості, залікові книжки студентів;
- модуль зворотнього зв’язку від студентів, який представлено опитуваннями та відгуками від здобувачів вищої освіти, щодо проведених занять, та відповідним аналізом результатів;
- фінансовий модуль, який дозволяє генерацію та завантаження фінансових звітів у вигляді розрахункових листів;
- модуль адміністрування, який надає користувачам з відповідним рівнем доступу, а саме ІТ-помічникам і адміністраторам функції керування основними частинами системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційна система управління вищим навчальним закладом як платформа реалізації управління академічним процесом. М. С. Львов, О. В. Співаковський, Д.Є. Щедролосьєв Д.Є

2. Львов М.С. Інтегроване програмне середовище вивчення курсу аналітичної геометрії для ВНЗ. Концепція, архітектура, функціональність /М.С.Львов//Наукові праці національного університету харчових технологій.–№ 30.– Київ:НУХТ, 2010. – С. 106 – 109.

3. Львов М.С. Шкільна система комп'ютерної алгебри ТерМ 7-9. Принципи побудови та особливості використання. / М.С.Львов // Науковий часопис НПУ ім.Драгоманова, серія №2. Комп'ютерно-орієнтовані системи навчання: зб.наук. праць/ редкол. –К.: НПУ ім.Драгоманова.–№3(10)–2005. с. 160–168.

4. Авторське свідоцтво 7668, МОНУ. Державний департамент інтелектуальної власності. «Програмно-методичний комплекс «Відеоінтерпретатор алгоритмів пошуку та сортування» освітньої галузі «Інформатика» для загальноосвітніх навчальних закладів / Львов М.С., Співаковський О.В., Зайцева Т.В., Кравцов Г.М., Кот С.М., Кравцов Д.Г., Герасименко К.С., Песчаненко В.С., Грабовський А.Ю., Хоруженко А.О.; заявл. 28.05.2003; Бюл. № 3. –2003.

5. Авторське свідоцтво 9524 МОНУ. Державний департамент інтелектуальної власності. Комп'ютерна програма «Програмне середовище «Системи лінійних рівнянь» освітньої галузі «Математика» для загальноосвітніх навчальних закладів. / Львов М.С., Співаковський О.В., Круглик В.С., Толстоп'ят К.В., Хоруженко А.О., Кравцов Г.М., Крекнін В.А., Кушнір Н.О. заявл. 09.03.2004. Бюл. №5. –2005.

6. Авторське свідоцтво 26159 МОНУ. Державний департамент інтелектуальної власності. Комп'ютерна програма «Web–Основи алгоритмізації та програмування» / Львов М.С., Співаковський О.В.,

Колеснікова Н.В., Ткачук І.М., Литвинов С.О.; заявл. 21.10.2008. Бюл. №17. – 008.

7. Авторське свідоцтво 32724 МОНУ. Державний департамент інтелектуальної власності. Комп'ютерна програма «Інтегроване середовище вивчення курсу «Аналітична геометрія» для вищих навчальних закладів» / Песчаненко В.С., Львов М.С., Співаковський О.В., Крекнін В.А., Саган О.В., Яцюта В.О., Лютіков В.А., Вінник М.О.; заявл. 06.04.10. Бюл. №21. – 2010.

8. Авторське свідоцтво 32723 МОНУ. Державний департамент інтелектуальної власності. Комп'ютерна програма «Педагогічний засіб навчального призначення «Алгебра, 8 клас» освітньої галузі математика для загальноосвітніх навчальних закладів» /Песчаненко В.С., Львов М.С., Співаковський О.В., Крекнін В.А., Шишко Л.С., Черненко І.Є., Грабовський А.Ю., Козловський Є.О., Вінник М.О.; заявл.06.04.10. Бюл. №21. – 2010.

9. Авторське свідоцтво 32722 МОНУ. Державний департамент інтелектуальної власності. Комп'ютерна програма «Педагогічний програмний засіб «Алгебра, 7 клас» освітньої галузі «Математика» для загальноосвітніх навчальних закладів/Песчаненко В.С., Львов М.С., Співаковський О.В., Крекнін В.А., Шишко Л.С., Черненко І.Є., Грабовський А.Ю., Вінник М.О.; заявл. 06.04.10. Бюл. №21. – 2010.

10. Кравцов Д. Г. Свідоцтво про реєстрацію авторського права на твір № 32719 Комп'ютерна програма «Система дистанційного навчання «Херсонський віртуальний університет» / Кравцов Д. Г., Кравцов Г. М., Співаковський О. В., Гнєдкова О. О., Камінська Н. Г. ; Міністерство освіти і науки України, Державний департамент інтелектуальної власності. – Київ. – 06.04.2010.

11. Методичні особливості побудови середовища дистанційного навчання «WEBALMIR» Круглик В.С. Journal of Information Technologies in Education (ITE), 088-093 – 2008.

12. Співаковський О.В., Лещинський О.Л. Навчальна програма “Світ лінійної алгебри” - Херсон: Вид-во Херсонського педагогічного університету, 2001. - 42 с.

13. Науково-дослідний інститут Прикладних інформаційних технологій [URL: http://www.ndipit.com.ua] (дата зверення 20.04.2020)

14. Цифрова трансформація держави [URL: http://www.kitsoft.com.ua] (дата зверення 30.05.2020)

15. Козин И.В. «Автоматизированная система «Деканат» / И.В. Козин, Т.В. Заховалко, С.В. Курапов. Вісник запорізького державного університету. 2003. № 1. С. 48–55.

16. АСУ «ВНЗ». Автоматизована система керування ВНЗ всіх рівнів акредитації. URL: [https://vuz.osvita.net/ua/asu-vnz/as-dekanat/] (дата зверення 19.10.2019)

17. Автоматизована система управління вищим навчальним закладом III – IV рівня акредитації. ТОВ «Юнітех+». [URL:http://www.unitex.com.ua/products/commercial-software/automated-system-for-highereducation-institution/] (дата зверення 19.10.2019)

18. Автоматизована система для планування навчального процесу. Інформаційно-обчислювальний центр забезпеченнянавчального процесу. URL: [https://ivc.kpi.ua] (дата зверення 19.10.2019)

19. Задорожна, Н.Т., Кузнецова, Т.В., СотниковаТ.Р (2006) Автоматизована база даних „Слухачі ЦІППО” як головний чинник комп’ютеризації діловодства навчального процесу Центрального інституту післядипломної педагогічної освіти In: П'ята міжнародна конференція Інтернет-освіта-наука - 2006. Збірник матеріалів. Т. 1. Універсум-Вінниця, м. Вінниця, стор. 44-46.

20. Інформаційна система «Електронний університет» url:[https://isu1.khmnu.edu.ua/isu/]. (дата зверення 19.10.2019)

21. Ю.М. Римар (2011) Створення автоматизованої системи управління навчальним процесом у вищих навчальних закладах

/Львівський інститут банківської справи Університету банківської справи, м. Київ/Науковий вісник НЛТУ України

22. EdApp microlearning platform; URL:[https://www.edapp.com/] (дата звернення 11.03.2021)

23. LearnWorlds, an online learning platform; URL:[https://www.learnworlds.com/] (дата звернення 11.03.2021)

24. V360E a virtual-reality-based online training software; URL: [https://home.v360e.com/] (дата звернення 11.03.2021)

25. WizIQ: all-in-one learning management system URL [https://www.wiziq.com/] (дата звернення 11.03.2021)

26. Virtual learning environment EdVance360 URL: [https://edvance360.com/lms/] (дата звернення 12.03.2021)

27. Virtual learning environment Wooclap URL:[https://www.wooclap.com/] (дата звернення 12.03.2021)

28. Virtual learning environment LearnCube URL: [https://www.learncube.com/] (дата звернення 13.03.2021)

29. Virtual learning environment Raven360 URL: [https://www.raven360.com/] (дата звернення 13.03.2021)

30. Britain, S. & Liber, O. (1999). *A Framework for Pedagogical Evaluation of Virtual Learning Environments*. JISC Technology Applications Programme (Report 41), 1999.

31. Weller, M. (2007). *Virtual Learning Environments: Using, Choosing and Developing Your VLE*. London: Routledge. ISBN 9780415414319.

32. Masterman, L. (2013). The challenge of teachers' design practice. Beetham, H. & Sharpe, R. (Eds.). *Rethinking Pedagogy in a Digital Age*. Oxford: Routledge. ISBN 9781351252805.

33. Edutechnica (2014). *LMS Date – The First Year Update*. Retrieved from https://edutechnica.com/2014/09/23/lms-data-the-first-year-update/ (date of access: 09.06.2020).

34. eLearning Industry (2020). *What Is a Virtual Classroom and Why Does It Matter?* Retrieved from https://elearningindustry.com/virtual-classroom-why-future-online-learning (date of access: 12.06.2020).

35. Husain, S. (2012). *Online communication between home and school. Case study: Improving the usability of the Unikum e-service in the primary schools of Tierp municipality.* Retrieved from https://www.diva-portal.org/smash/get/diva2:603927/FULLTEXT01.pdf (date of access: 12.07.2020).

36. Peat, M. (2000, July). Towards First Year Biology online: a virtual learning environment. *Educational Technology & Society*, *3 (3), 203–207*.

37. Xu, Y., Park, H., & Baek, Y. (October 2011). A New Approach Toward Digital Storytelling: An Activity Focused on Writing Self-efficacy in a Virtual Learning Environment. *Educational Technology & Society, 14 (4),* 181–191.

38. Everett, R. (2007). MLE Information Pack: Briefing Paper 1 – *MLEs and VLEs Explained*. JISC. Retrieved from http://www.jisc.ac.uk/uploaded_documents/bp1.pdf (date of access: 11.03.2021).

39. Virtual Learning Environments (VLE)». Accessed January 20, 2016. Retrieved 2018-09-24.

40. Padayachee I. (2002). *Intelligent Tutoring Systems: Architecture and Characteristics.*

41. Posey, G., Burgess, T., Eason, M. & Jones, Y. *Advantages and Disadvantages of the Virtual Classroom and the Role of the Teacher*. Retrieved from http://swdsi.org/swdsi2010/SW2010_Preceedings/papers/PA126.pdf (date of access: 11.09.2019).

42. Reese, S. (September 2015). Online learning environments in higher education: Connectivism vs. dissociation. *Education Information Technology*, 20(3), 579–588. doi:10.1007/s10639-013-9303-7.

43. Piccoli, G., Ahmad, R. & Ives, B. (December 2001). Web-Based Virtual Learning Environments: A Research Framework and a Preliminary Assessment of Effectiveness in Basic IT Skills Training. *MIS Quarterly*, 25 (4), 401–426. doi:10.2307/3250989.

44. Davis, C. (April 2014). Virtual Learning Rubric. Retrieved from http://vvv.de.mas.edu/odl/standards/Vlprubrits.pdf (date of access: 12.12.2020).

45. Walker, S. (2005). Development and Validation of an Instrument for Assessing Distance Education Learning Environments in Higher Education: The Distance Education Learning Environments Survey (DELES). *Learning Environment*, 8, 289-308. doi: 10.1007/s10984-005-1568-3.

46. Harnish, D. & Reeves, P. (2000). Issues in the Evaluation of Large-Scale Two-Way Interactive Distance Learning Systems. *International Journal of Educational Telecommunications*, 6 (3), 267–81.

47. Cook, D. (2007). Web-based learning: pros, cons and controversies. *Clinical Medicine*, 7 (1), 37-42. doi:10.7861/clinmedicine.7-1-37.

48. Demian, P. & Morris, J. (2015). The Use of Virtual Learning Environments and Their Impact on Academic Performance. *Engineering Education*, 7, 11–19. doi:10.11120/ened.2012.07010011.

49. Hodge, E. & Kibbe, S. (2010). *Collaborative Efforts: Teaching and Learning in Virtual Worlds*. Retrieved from https://er.educause.edu/articles/2010/6/collaborative-efforts-teaching-and-learning-in-virtual-worlds (date of access: 19.12.2021).

50. Morgan, G. (2003). *Features of Curse Management Systems*. Retrieved November 27, 2005.

51. Paulsen, M. (2003). *Online Education and Learning Management Systems – Global Elearning in a Scandinavian Perspective*. Oslo: NOK Forlaget.

52. Popat, C. (2007). *Virtuals There: Learning Platforms*. Scunthorpe: Yorkshire and the Humber Grid for Learning. ISBN 9780955600609.

53. Barton, D. & Gillen, J. (2010). A Research Briefing by the Technology Engaged Learning Phase of the Teaching and Learning Research Program. *Digital Literacies.* ISBN 978-0-85473-902-8

54. Sheey, K., Ferguson, R. & Clug, G. (2010). *Virtual Worlds: Controversies at the Frontier of Education*. Hauppauge, NI: Nova Science Publishers.

55. Ryann, K. (2009). Field Guide to Learning Management. *ASTD Learning Circles.*

56. Davis, B., Tsarmean, C. & Wagner, E. (2009). The Evolution of the LMS: From Management to Learning. *The Elearning Gould Research*, 24.

57. Raza, S., Qazi, W., Khan, K. & Salam, J. (April 2021). Social Isolation and Acceptance of the Learning Management System (LMS) in the time of COVID-19 Pandemic: An Expansion of the UTAUT Model. *Journal of Educational Computing Research,* 59 (2), 183–208.

58. Phillipo, J. (June 27, 2018). *LMS: The Missing Link and Great Enabler*.

59. Ohlsson, S. (1996) Learning from Performance Errors. Psychological Review, 103, 241-262.

60. Ohlsson, S. (1992) Constraint-based Student Modeling. Artificial Intelligence in Education, 3(4), 429-447.

61. Aldahwan, N. & Alsaeed, N. (August 2020). Use of Artificial Intelligent in Learning Management System (LMS): A Systematic Literature Review. *International Journal of Computer Applications,* 175 (August 2020),16–26. doi:10.5120/ijca2020920611.

62. *A Brief History of Online Education* (2019). Retrieved from https://warrington.ufl.edu/ (date of access: 01.02.2020).

63. GoDistanceLearning (2019) *History of Distance Learning*. Retrieved from www.godistancelearning.com/history-of-distance-learning (date of access: 19.02.2020).

64. Hubatskova, S. (June 2015). History and Perspectives of Elearning. *Proceedings - Social and Behavioral Sciences*, 191, 1187–1190.

65. Forster, E. (2014). *The Machine Stops*. A Forster Book. ISBN-10: 1434442047.

66. Solomon, A. (2013). A Critical Understanding of Learning Management Systems. *A Quarterly Refereed Journal of Dialogues on Education*, 2 (4), 4-12. ISSN 2278- 2435.

67. Parr, J. & Fung, I. (3 October 2006). *A Review of the Literature on Computer-Assisted Learning, particularly Integrated Learning Systems, and Outcomes with Respect to Literacy and Numeracy*. New Zealand Ministry of Education.

68. Anderson, H.; Koedinger, M. (1997). «Intelligent tutoring goes to school in the Big City».*International Journal of Artificial Intelligence in Education*.8: 30–43.

69. Watson, W. (2007). An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become?. *TechTrends,* 51(2), 28–34. doi:10.1007/s11528-007-0023-y.

70. Ashok Sharma (May 2015). The History of Distance Learning and the LMS. *LH Online Learning Made Simple*.

71. Paulsen, M. & Rekkedal, T. (2001). *The NKI Internet College: A Review of 15 Years of Delivery of 10,000 Online Courses.* Retrieved from https://www.irrodl.org/index.php/irrodl/article/view/17/354 (date of access: 22.07.2022).

72. Berking, P. & Gallagher, S. (2016). *Choosing an LMS*. The ADL Initiative. Retrieved from https://adlnet.gov/publications/2016/11/Choosing-a-Learning-Management-System-LMS/ (date of access: 23.07.2022).

73. Lin, S. (2015). *SaaS Learning Management System: Is your LMS Truly SaaS?* Retrieved from https://elearningindustry.com/saas-learning-management-system-lms-truly-saas (date of access: 23.07.2022).

74. Long, P. (2004). *Learning Management Systems (LMS). Encyclopedia of Distributed Learning*. Thousand Oaks: SAGE Publications, Inc. pp. 291–293. doi:10.4135/9781412950596.n99. ISBN 9780761924517.

75. Wang, Q., Woo, H. L., Quek, C. L., Yang, Y. & Liu, M. (2011). Using the Facebook group as a learning management system: An exploratory study. *British Journal of Educational Technology,* 43 (3), 428–438. doi:10.1111/j.1467-8535.2011.01195.x. ISSN 0007-1013.

76. Chaiprasurt, C. & Esichaikul, V. (2013). Enhancing motivation in online courses with mobile communication tool support: A comparative study. *The International Review of Research in Open and Distance Learning,* 14 (3), 377–401. doi:10.19173/irrodl.v14i3.1416. ISSN 1492-3831.

77. Schoonenboom, J. (2014). Using an adapted, task-level technology acceptance model to explain why instructors in higher education intend to use some learning management system tools more than others. *Computers & Education*, 71, 247–256. doi:10.1016/j.compedu.2013.09.016. ISSN 0360-1315.

78. Bradley, V. M. (2021). Learning Management System (LMS) Use with Online Instruction. *International Journal of Technology in Education,* 4(1), 68–92. doi:10.46328/ijte.36. ISSN 2689-2758.

79. Jones, K. (2019). Learning analytics and higher education: a proposed model for establishing informed consent mechanisms to promote student privacy and autonomy. *International Journal of Educational Technology in Higher Education*.

80. e-Literate (2017). *Academic LMS Market Share: A view across four global regions*.

81. Shutler, K. (2018). *SCORM is dead – what are the alternatives to SCORM?* Retrieved from https://plumestudio.com/blog/scorm-is-stagnant-heres-what-to-use-instead (date of access: 23.07.2022).

82. Ellis, R. & Calvo, R.A. (2007). Minimum indicators to quality assure blended learning supported by learning management systems. *Journal of Educational Technology and Society.*

83. González, J. F., Rodríguez, M. C. & Llamas, M. (2009). Enhancing Reusability in learning management systems through the integration of third-party tools. *39th IEEE Frontiers in Education Conference*. pp. 1–6. doi:10.1109/FIE.2009.5350672. ISBN 978-1-4244-4715-2. S2CID 5467495.

84. Valuates Reports (2022). *Learning Management System (LMS) Market to Grow USD 40360 Million by 2028 at a CAGR of 17.1% | Valuates Reports*. Retrieved from www.prnewswire.com (date of access:01.02.2024).

85. Malavolta, I., Verdecchia, R., Filipovic, B., Bruntink, M. & Lago, P. (2018). How Maintainability Issues of Android Apps Evolve. *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME),* pp. 334–344.

86. Connolly, P. J. (2001). A standard for success. *InfoWorld,* 23(42), 57-58.

87. Gibbons, A. S., Nelson, J. M., & Richards, R. (2002). The nature and origin of instructional objects. In D. A. Wiley (Ed.), *The instructional use of learning objects.*

88. Gilhooly, K. (2001). Making e-learning effective. *Computerworld*, 35(29), 52-53.

89. Hodgins, H. W. (2002). The future of learning objects. In D. A. Wiley (Ed.), *The instructional use of learning objects*.

90. Wiley, D. (2002). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The instructional use of learning objects*.

91. Anshari, M., Almunawar, M., Lim, S. A. & Al-Mudimigh, A. (2019). *Customer relationship management and big data enabled: Personalization & customization of services*.

92. Shaw, R. (1991). *Computer-Aided Marketing & Selling*. Butterworth Heinemann. ISBN 978-0-7506-1707-9.

93. Hargrave, M. (2023). *Customer Relationship Management – CRM goes beyond just software*. Retrieved from https://www.investopedia.com/terms/c/customer_relation_management.asp (date of access: 01.02.2024).

94. Wingard, N. (2020) *CRM History: The Evolution of Better Customer Service.* Retrieved from https://www.streetdirectory.com/ (date of access: 23.09.2021).

95. Mukherjee, S. (2017). How to build a global company from a small town: The Zoho story. *Techseen*.

96. Lakshman, J. (2008). *Customer Relationship Management: A Strategic Approach.* Global India Publications. ISBN 9788190721127. Retrieved 8 June 2017.

97. Gartner (2010). *Gartner Announces Customer Relationship Management Summit 2009*. Retrieved from http://www.gartner.com/ (date of access: 29.03.2020).

98. Ellis, R. K. (2009). Field Guide to Learning Management, *ASTD Learning Circuits.*

99. Davis, B., Carmean, C., & Wagner, E. (2009). The Evolution of the LMS : From Management to Learning. *The ELearning Guild Research*, 24.

100. Buttle, Francis (2003). *Customer relationship management*. London: Routledge. ISBN 9781136412578.

101. Tavana, A. F., Fili, S., Tohidy, A., Vaghari, R. & Kakouie, S. (2013). Theoretical Models of Customer Relationship Management in

Organizations. *International Journal of Business and Behavioral Sciences,*3(11).

102. Reinartz, W., Krafft, M. & Hoyer, W. D. (2004). The Customer Relationship Management Process: Its Measurement and Impact on Performance. *Journal of Marketing Research*.

103. Leach, B. (2003). *Success of CRM systems hinges on the establishment of measurable benefits*. Pulp & Paper.

104. S.-T., Wu (2007). *Knowledge discovery using pattern taxonomy model in textmining*.

105. Mostow, J. & Beck, J. (2006). Some useful tactics to modify, map and minedata from intelligent tutors. *Natural Language Engineering*, 12 (2), 195–208.

106. Mohamad, S. K. & Tasir, Z. (2013). Educational data mining: A review. *Procedia-Social and Behavioral Sciences*, 97, 320–324.

107. Baker, R. et al. (2010). Data mining for education. *International encyclopedia of education*, 7, pp. 112–118.

108. Romero, C., Ventura, S. & De Bra, P. (2004). Knowledge discovery with genetic programming for providing feedback to courseware authors. *User Modeling and User-Adapted Interaction*, 14 (5), 425–464.

109. Raghavan, N. S. (2005). Data mining in e-commerce: A survey. *Sadhana*, 30 (2-3), 275–289.

110. Romero, C., Ventura, S., Pechenizkiy, M. & Baker, R. S. (2010). *Handbook of educational data mining.* CRC Press.

111. Davis, I. (2008). What Are the Benefits of MVC?. *Internet Alchemy*. Retrieved from https://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/ (date of access: 01.12.2019).

112. Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Person Education, ISBN 0-321-12742-0.

113. Django Documentation (n.d.). *Templates*. Retrieved from https://docs.djangoproject.com/en/5.0/topics/templates/ (date of access: 01.02.2024).

114. Buschmann, F. (1996). *Pattern-Oriented Software Architecture*.

115. Gamma, E. et al. (1994). *Design Patterns.*

116. Moore, D. et al. (2007) Professional Rich Internet Applications: Ajax and Beyond.

117. Leff, A. & Rayfield, J. T. (2001). Web-Application Development Using the Model/View/Controller Design Pattern. *IEEE Enterprise Distributed Object Computing Conference,* 118–127.

118. Krill, P. (2014). React: Making faster, smoother UIs for data-driven Web apps. *InfoWorld*.

119. Hemel, Z. (2013). *Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews*.

120. Dawson, C. (July 25, 2014). JavaScript's History and How it Led to ReactJS. *The New Stack*.

121. Chourasia, R. (2023). *Convert Class Component to Function (Arrow) Component – React*.

122. Larsen, J. (2021). *React Hooks in Action with Suspense and Concurrent Mode.* Manning. ISBN 978-1720043997.

123. Schwarzmüller, M. (2018). *React - The Complete Guide* (incl. Hooks, React Router and Redux). Packt Publishing.

124. Wieruch, R. (2020). *The Road to React*. Leanpub. ISBN 978-1720043997.

125. Dere, M. (2017). *How to integrate create-react-app with all the libraries you need to make a great app*.

126. Panchal, K. (2022). *Angular vs React Detailed Comparison*. SitePoint.

127. Hámori, F. (2022). *The History of React.js on a Timeline*. RisingStack.

128. Lardinois, F. (2017). Facebook announces React Fiber, a rewrite of its React library. *TechCrunch*.

129. Github (2017). React Fiber Architecture.

130. Baer, E. (2023). Chapter 1. What Is React? *What React Is and Why It Matters*. Retrieved from https://www.oreilly.com/library/view/what-react-is/9781491996744/ch01.html (date of access: 13.03.2024).

131. Krill, P. (2014). React: Making faster, smoother UIs for data-driven Web apps. *InfoWorld*.

132. Haque, M.A., Haque, S., Zeba, S.et al. (2023). Sustainable and efficient E-learning internet of things system through blockchain technology. *E-Learning and Digital Media*, doi 10.1177/20427530231156711.

133. Panda, S.K., Mishra, V., Dash, S.P. et al. (Eds.) (2023). An introduction to blockchain technology: recent trends. *Recent Advances in Blockchain Technology, Real-World Applications* (pp.1-24). doi:10.1007/978-3-031-22835-3_1

134. Sharkey-Toppen, T.P., Hoffman, T.C., K. McCamey, et al. () Blockchain in education: linking competency assessment with credentialing. *Blockchain in Healthcare* (pp. 165-179).

135. Stawicki, S. (Ed.), *Blockchain in Healthcare: from Disruption to Integration*. Springer, Cham.

136. Pu, S.Y. & Lam, J.S.L. (2023). The benefits of blockchain for digital certificates: a multiple case study analysis. *Technol. Soc.,* 72, Article 102176.

137. Bhaskar, P., Tiwari, C.K. & Joshi, A. (2020). Blockchain in education management: present and future applications. *Interact. Technol. Smart Educ*., 18 (1), 1-17.

138. Steiu, M.F. (2020). Blockchain in education: opportunities, applications, and challenges. *First Monday*, 25 (9).

139. Chaka, C. (2023). Fourth industrial revolution—a review of applications, prospects, and challenges for artificial intelligence, robotics and

blockchain in higher education. *Res. Pract. Technol. Enhanc. Learn. (RPTEL)*, 18, p. 2.

140. Mozumder, M.A.I., Athar, A., Armand, T.P.T. et al. (2023). Technological roadmap of the future trend of metaverse based on IoT, blockchain, and AI techniques in metaverse education. *Proceedings of the 2023 25th International Conference on Advanced Communication Technology (ICACT), IEEE*, pp. 1414-1423.

141. Ma, Y.& Fang, Y.M. (2020). Current status, issues, and challenges of blockchain applications in education. *International Journal of Emerging Technologies in Learning (IJET)*, 15 (12), 20-31.

142. Tejedor, S., Cervi, L., Pérez-Escoda, A. et al. (2021). Higher education response in the time of coronavirus: perceptions of teachers and students, and open innovation. *Journal of Open Innovation: Technology, Market, and Complexity,* 7 (1), p. 43.

143. Bore, N., Karumba, S., Mutahi, J. et al. (2017). Towards blockchain-enabled school information hub. *Proceedings of the Ninth International Conference on Information and Communication Technologies and Development (ICTD -17), ACM*, pp. 1-4.

144. Alam, T. & Benaida, M. (2020). Blockchain and internet of things in higher education. *Universal Journal of Educational Research*, 8, pp. 2164-2174.

145. Jha, R. (2023). Survey on blockchain technology and security facilities in online education. In Panda, S.K., Mishra, V., Dash, S.P. et al. (Eds.) *Recent Advances in Blockchain Technology: Real-World Applications*. Springer, Cham (pp. 131-154).

146. Bucea-Manea-Ţoniş, R., Martins, O.M.D. et al. (2021). Blockchain technology enhances sustainable higher education. *Sustainability*, 13 (22), Article 12347.

147. Kaner, C. (2006). Exploratory Testing. *Quality Assurance Institute Worldwide Annual Software Testing Conference.* Orlando, FL.

148. Kaner, C., Falk, J. & Nguyen, H. Q. (1999). *Testing Computer Software (2nd ed.).* New York: John Wiley and Sons.

149. Kolawa, A. & Huizinga, D. (2007). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. ISBN 978-0-470-04212-0.

150. Ramler, R., Kopetzky, T. & Platz, W. (2012). Combinatorial Test Design in the TOSCA Testsuite: Lessons Learned and Practical Implications. *IEEE Fifth International Conference on Software Testing and Validation (ICST).* Montreal, QC, Canada.

151. Sharma, B. (2016). Ardentia Technologies: Providing Cutting Edge Software Solutions and Comprehensive Testing Services. *CIO Review (India ed.)*.

152. Graham, D., Van Veenendaal, E. & Evans, I. (2008). Foundations of Software Testing. *Cengage Learning* (pp. 57–58).

153. Laycock, G. T. (1993). *The Theory and Practice of Specification Based Software Testing* (dissertation thesis). Department of Computer Science, University of Sheffield.

154. Lewis, W.E. (2016). Software Testing and Continuous Quality Improvement (3rd ed.). *CRC Press* (pp. 68–73).

155. Code Project (2006). SOA *Testing Tools for Black, White and Gray Box.* Crosscheck Networks. Retrieved from https://www.codeproject.com/Articles/14506/SOA-Testing-using-Black-White-Gray-Box-Techniques. (date of access: 28.07.2022).

156. Bourque, P., Fairley, R. E. (Eds.) (2014). Chapter 5. Guide to the Software Engineering Body of Knowledge. 3.0. *IEEE Computer Society*. ISBN 978-0-7695-5166-1.

157. Dooley, J. (2011). *Software Development and Professional Practice.* APress (pp. 193–204). ISBN 978-1-4302-3801-0.

158. Binder, R. V. (1999). Testing Object-Oriented Systems: Objects, Patterns, and Tools. *Addison-Wesley Professional*. p. 45. ISBN 978-0-201-80938-1.

159. Beizer, B. (1990). *Software Testing Techniques (Second ed.).* New York: Van Nostrand Reinhold. pp. 21, 430. ISBN 978-0-442-20672-7.

160. Woods, A. J. (2015). *Operational Acceptance – an application of the ISO 29119 Software Testing standard.* Capgemini Australia.

161. Kaner, C., Bach, J. & Pettichord, B.(2001). *Lessons Learned in Software Testing: A Context-Driven Approach*. Wiley (pp. 31–43). ISBN 978-0-471-08112-8.

162. Myers, G. (2004). Sandler, C; Badgett, T; Thomas, M. (eds.). *The Art of Software Testing (2 ed.).* Wiley. ISBN 9780471469124.

163. Ammann, P. & Offutt, J. (2008). *Introduction to Software Testing*. Cambridge University Press. (p. 215). ISBN 978-0-521-88038-1.

164. O'Reilly, T. (2005). What is Web 2.0. O'Reilly Media. Section 4. *End of the Software Release Cycle*.

165. Microsoft (2009). Test-Driven Development and Continuous Integration for Mobile Applications. Retrieved from msdn.microsoft.com.

166. Joshi, S. & Orso, A. (2007). SCARPE: A Technique and Tool for Selective Capture and Replay of Program Executions. *2007 IEEE International Conference on Software Maintenance,* 234–243.

167. Saieva, A., Singh, S. & Kaiser, G. (2020). Ad hoc Test Generation Through Binary Rewriting. *2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM).* Adelaide, Australia: IEEE. pp. 115–126.

168. Gotel, O. et al. (2012). Cleland-Huang, J., Gotel, O., Zisman, A. (eds.). *Software and Systems Traceability.* Springer London. doi:10.1007/978-1-4471-2239-5_1. ISBN 9781447122388.

169. Strom, D. (2009). We're All Part of the Story. *Software Test & Performance Collaborative.*

170. Willison, J. S. (2004). Agile Software Development for an Agile Force. *CrossTalk. STSC*.

171. Fewster, M. & Graham, D. (1999). *Software Test Automation*. Addison Wesley. ISBN 978-0-201-33140-0.

172. Griffiths, M. (2005). Teaching agile project management to the PMI. *Agile Development Conference (ADC'05)*. pp. 318–322.

173. Bossavit, L. (2013). *The Leprechauns of Software Engineering*. Leanpub.

174. Kucharski, M. (2011). *Making Unit Testing Practical for Embedded Development*.

175. Klabnik, S. & Nichols, C. (2015–2023) with contributions from the Rust Community (2015–2023).

176. Python Documentation (2016). *Unittest – Unit testing framework.* Retrieved from https://docs.python.org/3/library/unittest.html. (date of access: 29.10.2020).

177. Copeland, L. (2001). *Extreme Programming*. Computerworld.

178. Beck, K. (2002). *Test-Driven Development by Example*. Vaseem: Addison Wesley. ISBN 978-0-321-14653-3.

179. Agile Test Driven Development». Agile Sherpa. 2010-08-03. Archived from the original on 2012-07-23.

180. Papis, B, Grochowski, K., Subzda, K. & Sijko, K. (2020). Experimental evaluation of test-driven development with interns working on a real industrial project. *IEEE Transactions on Software Engineering 2020*. DOI: 10.1109/TSE.2020.3027522

181. Daniel Chen (2017). *Pandas for Everyone: Python Data Analysis*. Addison-Wesley Professional. ISBN 978-0134546933.

182. Wes McKinney (2011). «pandas: a Foundational Python Library for Data Analysis and Statistics» (PDF). Retrieved 2 August 2018

183. Spivakovsky, O., Omelchuk, S., Malchykova, D., Tsapiv, A. & Lemeshchuk, O. Academic solidarity and digitization: Management of a

displaced university. *Probl. Perspect. Manag.* **21**, 40–51. https://doi.org/10.21511/ppm.21(2-si).2023.06 (2023).

184. Львов, М. С., & Лемещук О. І. (2023). Задача на найкоротший шлях у графі для виконавця з обмеженими ресурсами. Науковий вісник Івано-Франківського національного технічного університету нафти і газу, (2(55), 47–53. https://doi.org/10.31471/1993-9965-2023-2(55)-47-53.

185. Yasinska-Damri, L.; Babichev, S.; Spivakovsky, A.; Lemeshchuk, O. Formation and Analysis of Gene Expression Data Based on the Joint Use of Data Mining and Machine Learning Techniques. In Proceedings of the CEUR Workshop Proceeding, IntelITSIS'2023: 4th International Workshop on Intelligent Information Technologies and Systems of Information Security, Khmelnytskyi, Ukraine, 22–24 March 2023; Volume 3373, pp. 87–98.

186. Spivakovsky, A., Petukhova, L., Poltoratskyi, M., Lemeshchuk, O., Volianiuk, A., Kazannikova, O., ... & Chepurna, S. (2023, September). Theoretical Principles of Measuring and Interpreting Levels of Attention, Involvement and Organizing Feedback of Students to the Educational Process Using Automated Software Products. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 144-159). Cham: Springer Nature Switzerland.

187. Програмний комплекс ЄДЕБО [URL:https://www.inforesurs.gov.ua/wp-content/uploads/2019/10/kk_r1_zaklad_osvity.pdf] (дата звернення 22.11.2021)

ДОДАТКИ

ДОДАТОК А

Відгуки щодо занять

Факультет психології, історії та соціології

Проведено занять викладачами	4671
Пройдено опитувань студентами	17808
Наскільки був зрозумілий для вас навчальний матеріал?	4,869
Наскільки вас зацікавила тематика і зміст заняття?	4,873
Наскільки ви задоволені атмосферою на занятті?	4,884
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,891
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,865

Факультет української й іноземної філології та ж...

Проведено занять викладачами	3737
Пройдено опитувань студентами	15601
Наскільки був зрозумілий для вас навчальний матеріал?	4,867
Наскільки вас зацікавила тематика і зміст заняття?	4,879
Ставлення до студентів	5,000
Наскільки ви задоволені атмосферою на занятті?	4,887
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,897
Об'єктивність оцінювання здобувачів	5,000
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,874
Якість викладання	5,000
Мотивування здобувачів	5,000
Мовленнєва поведінка викладача	5,000

Факультет фізичного виховання та спорту

Проведено занять викладачами	2336
Пройдено опитувань студентами	9491
Мовленнєва поведінка викладача	5,000
Мотивування здобувачів	4,714
Наскільки був зрозумілий для вас навчальний матеріал?	4,851
Наскільки вас зацікавила тематика і зміст заняття?	4,859
Наскільки ви задоволені атмосферою на занятті?	4,881
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,839
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,872
Об'єктивність оцінювання здобувачів	3,714
Ставлення до студентів	5,000
Якість викладання	4,857

Факультет біології, географії та екології

Проведено занять викладачами	2117
Пройдено опитувань студентами	8082
Наскільки був зрозумілий для вас навчальний матеріал?	4,833
Наскільки вас зацікавила тематика і зміст заняття?	4,849
Наскільки ви задоволені атмосферою на занятті?	4,868
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,848
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,866

Факультет комп'ютерних наук, фізики та ма...

Проведено занять викладачами	2270
Пройдено опитувань студентами	7014
Наскільки був зрозумілий для вас навчальний матеріал?	4,809
Наскільки вас зацікавила тематика і зміст заняття?	4,813
Наскільки ви задоволені атмосферою на занятті?	4,825
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,850
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,813

Факультет культури і мистецтв

Проведено занять викладачами	1987
Пройдено опитувань студентами	5845
Наскільки був зрозумілий для вас навчальний матеріал?	4,819
Наскільки вас зацікавила тематика і зміст заняття?	4,838
Наскільки ви задоволені атмосферою на занятті?	4,831
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,806
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,834

Медичний факультет

Проведено занять викладачами	4193
Пройдено опитувань студентами	9594
Наскільки був зрозумілий для вас навчальний матеріал?	4,879
Наскільки вас зацікавила тематика і зміст заняття?	4,905
Наскільки ви задоволені атмосферою на занятті?	4,919
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,895
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,923

Педагогічний факультет

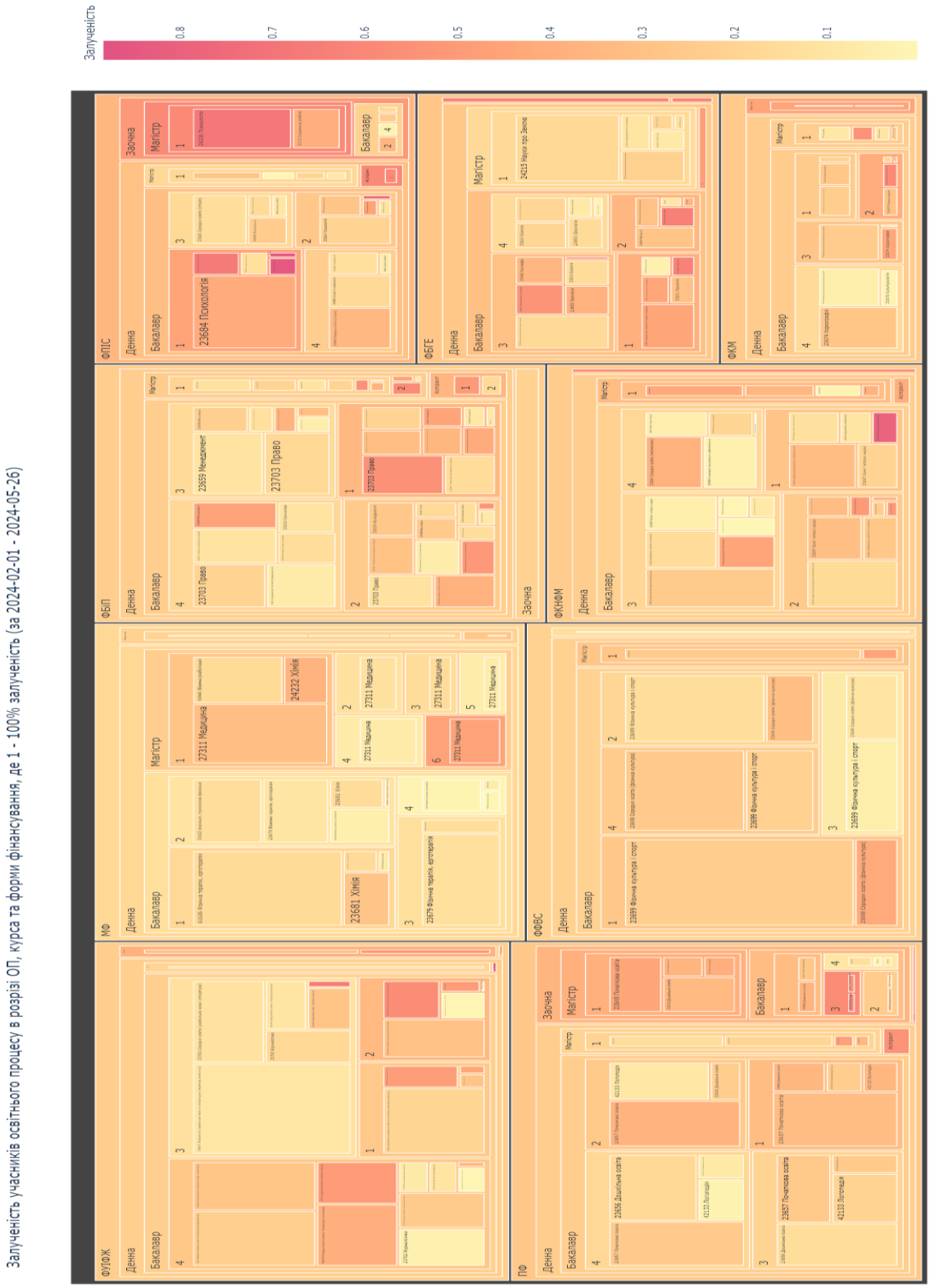
Проведено занять викладачами	3679
Пройдено опитувань студентами	16358
Наскільки був зрозумілий для вас навчальний матеріал?	4,877
Наскільки вас зацікавила тематика і зміст заняття?	4,877
Наскільки ви задоволені атмосферою на занятті?	4,892
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,871
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,892

Факультет бізнесу і права

Проведено занять викладачами	3273
Пройдено опитувань студентами	10510
Мовленнєва поведінка викладача	5,000
Мотивування здобувачів	5,000
Наскільки був зрозумілий для вас навчальний матеріал?	4,848
Наскільки вас зацікавила тематика і зміст заняття?	4,846
Наскільки ви задоволені атмосферою на занятті?	4,860
Наскільки для вас є дієвими підходи активізації пізнавального інтересу й мисленнєвої діяльності?	4,836
Наскільки зручно для вас налагоджено зворотний зв'язок та комунікацію з викладачем?	4,862
Об'єктивність оцінювання здобувачів	4,500
Ставлення до студентів	5,000
Якість викладання	5,000

Додаток Б

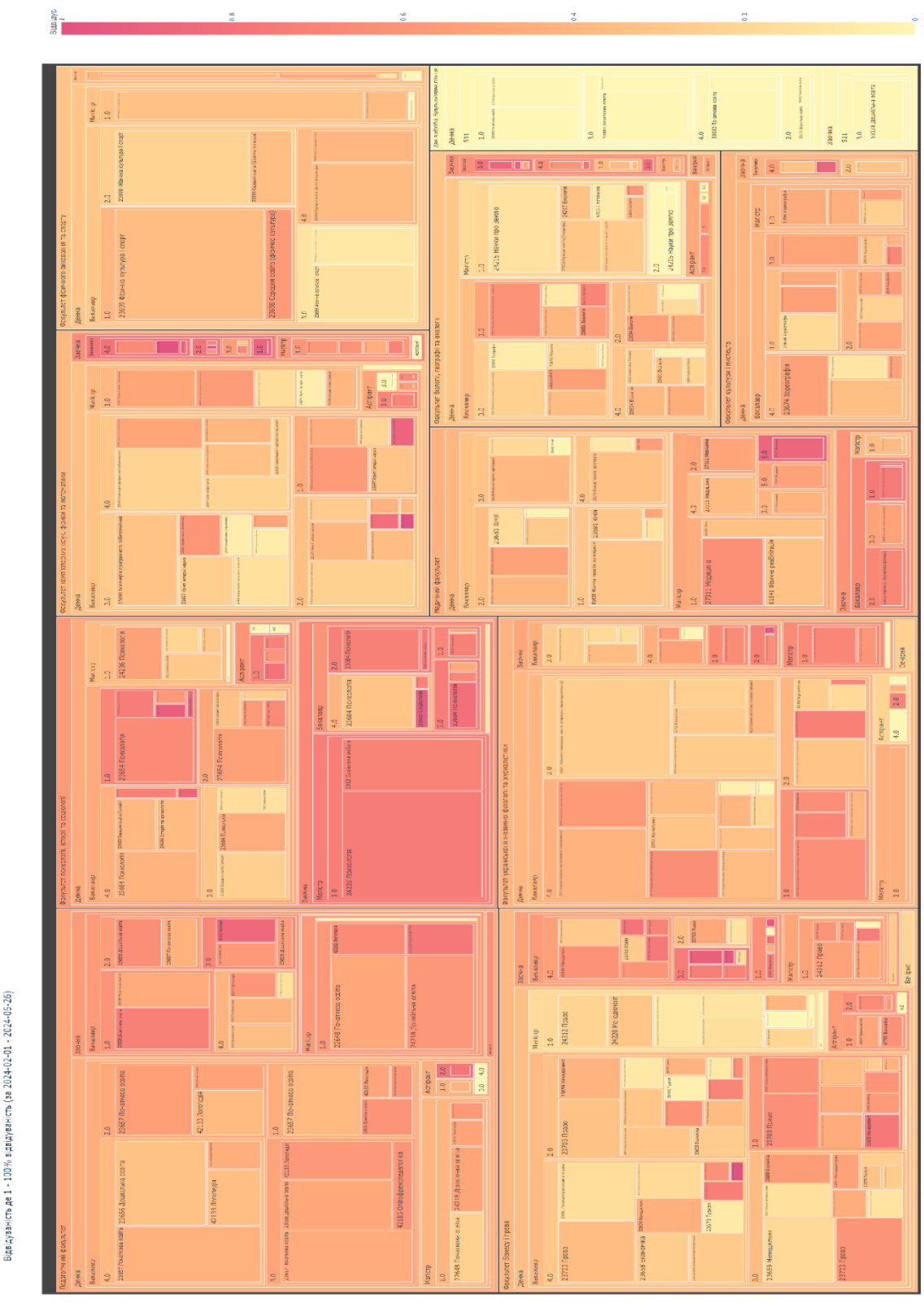
Карта залученості до опитувань після занять



Залученість учасників освітнього процесу в розрізі ОП, курса та форми фінансування, де 1 - 100% залученість (за 2024-02-01 - 2024-05-26)

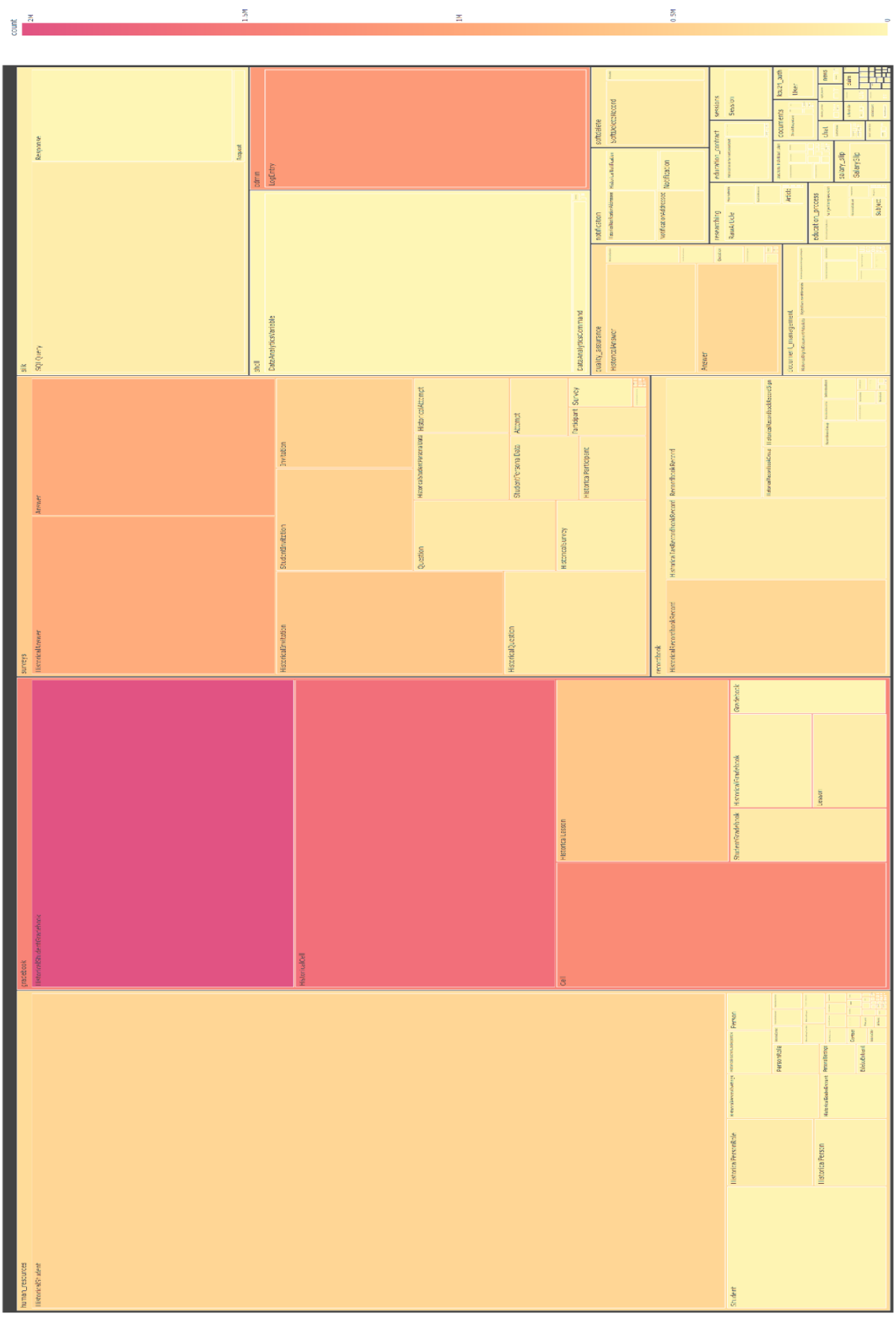
Додаток В

Картва відвідуваності навчальних занять



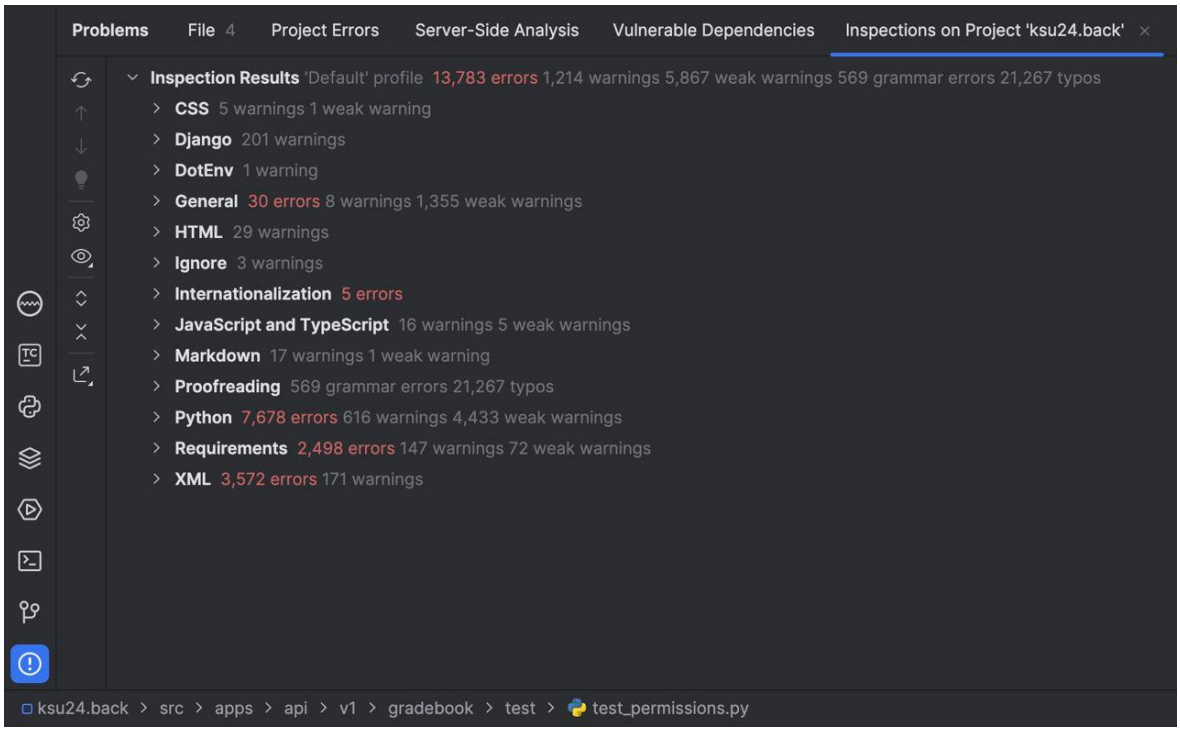
Додаток Г

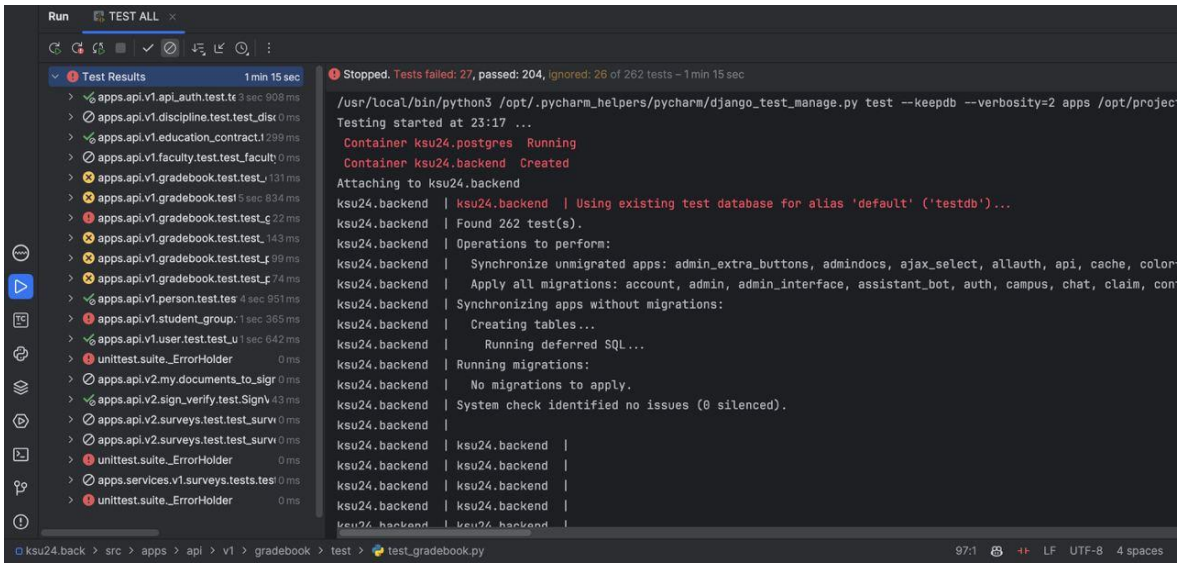
Карта кількості даних збережених у базу даних



Додаток Д

Інструмент Sonar Lint





Додаток Е

Редактор коду з прикладом графіку