

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА (проект)

на здобуття ступення вищої освіти «магістр»

на тему:

**«Створення програмної системи підтримки проведення лабораторних
робіт з програмування»**

Виконав: здобувач 2курсу, 12-231М групи

Спеціальності 122 комп'ютерні науки
другого (магістерського) рівня освіти

Шуляр Сергій Романович

Керівник: Львов М.С., доктор фізико
математичних наук, професор

Рецензент: Огнєва О.Є. кандидатка технічних
наук, доцентка, завідувачка кафедри
програмних засобів і технологій Херсонський
національно технічний університет

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1. Теоретичні засади автоматизації лабораторних робіт та аналіз існуючих систем.....	8
1.1. Проблематика організації лабораторних робіт у сучасному освітньому середовищі.	8
1.2. Необхідність автоматизації оцінювання та контролю.	10
1.3. Аналіз можливостей наявних систем (Moodle, Google Classroom, інші).....	14
1.4. Обмеження існуючих рішень у контексті дисциплін програмування.	16
1.5. Обґрунтування вибору підходу та вимог до нової системи	18
РОЗДІЛ 2. Технологічні засади та проектування програмної системи взаємодії з KSU24 і KSU Online.	26
2.1. Обґрунтування вибору технологій (React, Node.js, Express, SQLite).....	26
2.2. Архітектура системи та модель інтеграції з KSU24 і KSU Online (Moodle).....	29
2.3. API-орієнтований підхід: структура та принципи взаємодії компонентів.....	37

2.4. Проектування бази даних та логіки оцінювання.....	50
2.5. Забезпечення безпеки, масштабованості та коректної взаємодії зі сторонніми сервісами.....	54
РОЗДІЛ 3. Реалізація програмної системи та оцінювання її ефективності	
3.1. Реалізація фронтенд-модуля.....	59
3.2. Реалізація бекенд-модуля та серверної логіки.....	61
3.3. Інтеграція з KSU24 та KSU Online через REST API.....	63
3.4. Тестування функціоналу, інтерфейсу та взаємодії компонентів.....	64
3.5. Аналіз результатів, виявлені недоліки та перспективи розвитку системи.....	66
ВИСНОВКИ	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТКИ.....	80
Додаток А. Структура бази даних	80
Додаток Б. Фрагменти фронтенд-коду	83
Додаток В. Скріншоти інтерфейсу	89
Додаток Г. Фрагменти бекенд-коду	90
Додаток Д. Інструкція користувача	94

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Позначення	Розшифрування
API	Application Programming Interface — інтерфейс прикладного програмування
UI	User Interface — інтерфейс користувача
UX	User Experience — досвід користувача
REST	Representational State Transfer — архітектурний стиль веб-сервісів
CRUD	Create, Read, Update, Delete — базові операції даними
JSON	JavaScript Object Notation — формат обміну даними
SQL	Structured Query Language — мова запитів до даних
SQLite	Легка реляційна база даних, що зберігається у файлі
React	JavaScript-бібліотека для побудови інтерфейсів
Express	Фреймворк для Node.js для створення веб-серверів

ВСТУП

Актуальність теми зумовлена необхідністю підвищення ефективності організації лабораторних робіт з програмування в умовах інтенсивного розвитку дистанційних та змішаних форм навчання. У сучасному освітньому просторі лабораторні заняття відіграють ключову роль у формуванні професійних компетентностей здобувачів вищої освіти ІТ-спеціальностей, адже саме вони забезпечують практичне застосування теоретичних знань. Зростання кількості здобувачів вищої освіти, потреба в оперативному контролі успішності та важливість прозорого оцінювання створюють потребу у спеціалізованих програмних рішеннях, здатних автоматизувати обробку лабораторних робіт, підрахунок балів та формування допусків до підсумкового контролю. Існуючі універсальні платформи, зокрема Moodle чи Google Classroom, хоча й забезпечують базову підтримку освітнього процесу, не надають достатньої гнучкості в налаштуванні логіки оцінювання, що є критично важливим саме для дисциплін з програмування. Це посилює потребу у створенні програмної системи, орієнтованої на специфічні вимоги таких курсів.

Зв'язок роботи з науковими програмами, планами, темами. Виконання роботи обумовлене сучасними тенденціями цифрової трансформації освіти, що відображені в державних програмах інформатизації та модернізації освітнього середовища. Тематика дослідження узгоджується з науковими планами кафедри програмного забезпечення, яка здійснює дослідження у галузі розробки цифрових платформ для підтримки навчання і контролю знань. Таким чином, робота відповідає стратегічним напрямам наукової діяльності підрозділу та спрямована на вдосконалення цифрових інструментів освітнього процесу.

Метою дослідження є розроблення програмної системи підтримки лабораторних робіт з програмування, яка забезпечує автоматизацію завантаження робіт, їх оцінювання за шкалою 0–5 балів, розрахунок підсумкових показників та формування допуску до заліку.

Згідно до поставленої мети були сформульовані такі **завдання**:

1. Проаналізувати існуючі програмні рішення для організації та контролю лабораторних робіт.
2. Визначити функціональні та нефункціональні вимоги до програмної системи.
3. Розробити архітектуру системи з урахуванням вимог до масштабованості та модульності.
4. Реалізувати фронтенд частину з використанням React та Material UI.
5. Реалізувати бекенд на основі Node.js та Express із підтримкою логіки оцінювання.
6. Створити структуровану базу даних SQLite для збереження результатів студентів.
7. Забезпечити інтеграцію компонентів через REST API.
8. Провести тестування програмного продукту та оцінити його ефективність у практичному використанні.

Об'єктом дослідження є процес організації та оцінювання лабораторних робіт з програмування в закладах вищої освіти.

Предметом дослідження є програмна система, призначена для підтримки виконання, перевірки та аналізу лабораторних робіт здобувачів вищої освіти.

У дослідженні використано комплекс **методів**, до яких належать:

- аналіз літературних джерел та програмних платформ;
- моделювання архітектури програмної системи;

- проектування бази даних;
- розроблення програмного забезпечення із застосуванням сучасних фреймворків;
- тестування функціональних можливостей системи; порівняльний аналіз результатів.

Наукова новизна полягає у формуванні структурованої моделі оцінювання лабораторних робіт за числовою шкалою (0–5) та розробленні алгоритму автоматичного обчислення підсумкових балів і допуску до підсумкового контролю. Запропонована інтеграція фронтенд- та бекенд-компонентів через REST API забезпечує можливість подальшого масштабування та розширення функціоналу.

Практичне значення одержаних результатів полягає у створенні програмної системи, яка може бути впроваджена в освітній процес закладів вищої освіти з метою автоматизації контролю лабораторних робіт, зменшення навантаження на викладачів та підвищення прозорості оцінювання. Система є адаптивною та може бути використана для інших дисциплін, що потребують поетапного оцінювання.

Результати дослідження були **апробовані** під час внутрішнього тестування на кафедрі комп'ютерних наук та програмної інженерії, що дало змогу оцінити їх ефективність та отримати практичні рекомендації щодо вдосконалення функціоналу.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ АВТОМАТИЗАЦІЇ ЛАБОРАТОРНИХ РОБІТ ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

1.1. Проблематика організації лабораторних робіт у сучасному освітньому середовищі.

У сучасній системі вищої освіти лабораторні роботи залишаються ключовим інструментом формування практичних компетентностей здобувача вищої освіти. Особливо це стосується галузей, пов'язаних з інформаційними технологіями, де ефективність засвоєння матеріалу безпосередньо залежить від здатності здобувача застосовувати теоретичні знання у практичних ситуаціях. Проте, із зростанням ролі дистанційних і змішаних форм навчання організація лабораторних занять зазнає суттєвих змін, що породжує низку проблем, пов'язаних з контролем, оцінюванням та забезпеченням прозорості навчального процесу [1][2]. Однією з найбільш помітних проблем є збільшення кількості здобувачів вищої освіти у поєднанні з ускладненням змісту освітніх програм. Це створює підвищене навантаження на науково-педагогічних працівників, які змушені опрацьовувати значні масиви лабораторних робіт, перевіряти коректність виконання, виставляти оцінки, надавати рекомендації та фіксувати результати у різних навчальних платформах. За умов традиційного підходу, що передбачає переважно ручну перевірку, такі процеси стають не лише трудомісткими, але й схильними до суб'єктивності, помилок та затримок у зворотному зв'язку [3][4].

Ситуація ускладнюється ще й тим, що в умовах військового стану та нестабільності частина освітнього процесу відбувається дистанційно. Це вимагає стабільної цифрової інфраструктури, яка дозволяє здобувачам вищої освіти своєчасно виконувати завдання, а науково-педагогічним працівникам — здійснювати контроль незалежно від обставин. Водночас якість організації

лабораторних занять суттєво залежить від того, наскільки ефективно інтегровані цифрові системи у загальну модель навчального процесу закладу вищої освіти [5][6].

Ще однією проблемою є відсутність єдиних стандартів оцінювання лабораторних робіт у різних програмних дисциплінах. Часто кожен науково-педагогічний працівник формує власні критерії, шкали та підходи до контролю, що ускладнює порівнянність результатів та знижує рівень внутрішньої узгодженості освітнього процесу. В умовах великої кількості здобувачів і різноманіття дисциплін це призводить до фрагментації підходів та ускладнює управління навчальним процесом на рівні закладу вищої освіти [7][8].

Паралельно із зазначеними викликами зростає потреба у прозорості навчального процесу. Здобувачі вищої освіти очікують можливості відстежувати власний прогрес, отримувати оперативний зворотний зв'язок і мати чітке розуміння того, як формується підсумковий бал. Проте у більшості традиційних підходів механізми зворотного зв'язку залишаються нерегулярними та фрагментарними, оскільки науково-педагогічні працівники не мають інструментів для швидкого і стандартизованого коментування виконаних робіт [9][10].

У цьому контексті важливо враховувати й технічний аспект організації лабораторних занять. Частина завдань, особливо у дисциплінах програмування, передбачає завантаження файлів, тестових даних, скріншотів або вихідного коду. Відсутність спеціалізованого програмного забезпечення змушує науково-педагогічних працівників використовувати різні сторонні інструменти (електронну пошту, хмарні сховища, месенджери), що ускладнює систематизацію даних та зменшує ефективність оцінювання [11][12].

Більшість закладів вищої освіти частково вирішують ці проблеми за допомогою систем дистанційного навчання, таких як Moodle. Однак їх функціонал не завжди достатньо гнучкий для побудови адаптивної логіки оцінювання або організації індивідуальних траєкторій лабораторних робіт, що особливо актуально для дисциплін програмування [13][14]. Здобувачі вищої освіти часто стикаються з труднощами під час завантаження робіт або отримання зворотного зв'язку, а науково-педагогічні працівники — із необхідністю дублювати дані в різних системах або формувати оцінки вручну.

Поступове ускладнення освітніх процесів також підкреслює потребу у забезпеченні формальної перевірки допуску до заліку чи екзамену, що базується на накопичених результатах лабораторних робіт. У традиційній моделі цей процес нерідко відбувається вручну, що створює додатковий ризик виникнення неточностей і потребує додаткового часу [15], [16]. Узагальнюючи, проблематика організації лабораторних занять у сучасному цифровізованому освітньому середовищі полягає не лише у збільшенні обсягів завдань чи складності контролю, а у відсутності цілісної інфраструктури, яка б забезпечувала автоматизоване оцінювання, прозорість, стандартизованість та ефективну взаємодію всіх суб'єктів навчального процесу. Ці чинники і визначають необхідність розробки спеціалізованої програмної системи, здатної інтегруватися з наявними платформами закладу вищої освіти та доповнювати їх функціональні можливості, забезпечуючи як технологічну, так і педагогічну цілісність процесу.

1.2. Необхідність автоматизації оцінювання та контролю.

У сучасних закладах вищої освіти процес оцінювання навчальних досягнень здобувачів є невід'ємною складовою організації освітнього

процесу, оскільки від його ефективності залежить не лише якість навчання, а й формування професійних компетентностей, що відповідають державним стандартам освіти та міжнародним вимогам [17][18]. Особливо це стосується дисциплін, пов'язаних із програмуванням, розробкою програмного забезпечення та інформаційними технологіями, де навчальні завдання відзначаються складною структурою та поетапністю виконання. У таких курсах оцінювання включає як оцінку практичних результатів, так і формалізовану перевірку знань та навичок здобувачів вищої освіти, що передбачає одночасну роботу з великим обсягом інформації, тестових та проектних завдань.

Традиційні підходи до оцінювання лабораторних робіт у більшості закладів вищої освіти передбачають ручну перевірку виконаних завдань, виставлення оцінок на основі формальних критеріїв та додаткове документування результатів. Це створює значне навантаження на науково-педагогічного працівника, який відповідає не лише за точність оцінювання, але й за своєчасність надання зворотного зв'язку, контроль за виконанням завдань, а також за підготовку підсумкових балів для формування допуску здобувачів вищої освіти до підсумкового контролю [19][20]. Така модель оцінювання стає особливо складною та мало ефективною при збільшенні чисельності здобувачів, адже ручна перевірка великої кількості робіт збільшує ризик помилок, затримки у зворотному зв'язку та суб'єктивного впливу на оцінювання, що негативно відображається на об'єктивності та прозорості процесу [21][22].

Крім того, традиційні методи не дозволяють реалізувати поетапний контроль знань, коли кожен крок виконання завдання оцінюється окремо, а проміжні результати автоматично інтегруються у підсумковий бал. Такий підхід особливо важливий для дисциплін програмування, де завдання часто

містять декілька логічних етапів, включаючи написання коду, його тестування, оптимізацію алгоритмів та документацію [23]. Відсутність автоматизації призводить до того, що науково-педагогічний працівник змушений витратити значну кількість часу на однотипні операції, що знижує ефективність навчального процесу та ускладнює впровадження сучасних педагогічних технологій, спрямованих на індивідуалізацію та персоналізацію навчання.

Ще одним важливим аспектом є обмежена можливість інтеграції з іншими інформаційними системами закладу вищої освіти та зовнішніми LMS, наприклад KSU24 і KSU Online на базі Moodle. У сучасному освітньому середовищі здобувачі вищої освіти можуть брати участь у дистанційних та змішаних формах навчання, де інформація про прогрес у навчанні, результати лабораторних робіт та проміжні оцінки має своєчасно передаватися між системами. Недостатня інтеграція створює дублювання даних, затримки в оновленні результатів та додаткове навантаження на науково-педагогічних працівників, які змушені контролювати та узгоджувати інформацію вручну [24][25].

Узагальнюючи, можна зазначити, що традиційна модель оцінювання лабораторних робіт у дисциплінах програмування є ресурсомістких, малоефективною та не достатньо прозорою. Вона не дозволяє забезпечити швидкий зворотний зв'язок, ефективний моніторинг індивідуального прогресу здобувачів вищої освіти, а також інтеграцію з іншими системами, що є критично важливим для сучасного цифрового освітнього середовища [26][27]. Ці фактори підтверджують актуальність розробки спеціалізованої програмної системи, яка б автоматизувала оцінювання, інтегрувалася з наявними платформами, забезпечувала безпечне зберігання даних та підтримувала побудову індивідуальних навчальних траєкторій.

Одним із ключових аспектів, що обумовлює необхідність автоматизації, є забезпечення стандартизованого підходу до оцінювання. Різні науково-педагогічні працівники застосовують власні шкали оцінювання, критерії та методи підрахунку балів, що ускладнює порівняння результатів між різними групами здобувачів та знижує загальний рівень узгодженості освітнього процесу [21]. Автоматизована система дозволяє формалізувати правила оцінювання, забезпечити їхню єдність і прозорість, а також зменшити вплив людського чинника на підсумковий результат.

Додатково, автоматизація оцінювання створює можливості для швидкого формування аналітичних даних щодо прогресу здобувачів вищої освіти. Це включає моніторинг виконання завдань, визначення середнього балу за лабораторні роботи, контроль за пропущеними чи неякісно виконаними завданнями та автоматичне формування допусків до заліків чи іспитів [22]. Наявність таких інструментів дозволяє науково-педагогічним працівникам ефективно планувати навчальні заняття, своєчасно втручатися у процес при виявленні проблем та надавати здобувачам вищої освіти оперативний зворотний зв'язок.

Особливо актуальним стає питання автоматизації у контексті дистанційного навчання та змішаних форм організації освітнього процесу. У цих умовах здобувачі вищої освіти можуть виконувати лабораторні роботи поза межами закладу вищої освіти, а науково-педагогічні працівники — контролювати виконання завдань у режимі онлайн. Відповідно, потреба у цифрових інструментах, що забезпечують безперервний моніторинг і оцінювання, стає критичною [23]. Така автоматизація забезпечує оперативність і точність перевірки, мінімізує адміністративне навантаження на персонал і сприяє підвищенню задоволеності здобувачів освітнім процесом.

Ще одним важливим аспектом є інтеграція систем оцінювання з іншими компонентами навчального середовища закладу вищої освіти. Наприклад, автоматизована система оцінювання має бути здатна взаємодіяти з платформами дистанційного навчання, такими як KSU Online на базі Moodle, та внутрішніми системами закладу, наприклад KSU24. Така інтеграція дозволяє об'єднати дані про результати лабораторних робіт, успішність здобувачів та їх участь у навчальному процесі в єдиному середовищі, забезпечуючи як технологічну, так і педагогічну цілісність процесу [24].

Нарешті, автоматизація оцінювання сприяє підвищенню прозорості та об'єктивності навчального процесу. Завдяки формалізованим алгоритмам перевірки та підрахунку балів кожен здобувач вищої освіти отримує чітке уявлення про власний прогрес і причини формування підсумкових результатів. Це, в свою чергу, стимулює мотивацію до навчання, зменшує ризик конфліктів та забезпечує довіру до системи оцінювання з боку здобувачів і адміністрації закладу [25].

Узагальнюючи, необхідність автоматизації оцінювання та контролю лабораторних робіт обумовлена комплексом педагогічних, організаційних і технологічних факторів. Вона дозволяє оптимізувати навчальний процес, забезпечити стандартизованість оцінювання, зменшити адміністративне навантаження на науково-педагогічних працівників, а також інтегрувати дані в єдину цифрову інфраструктуру закладу вищої освіти, що особливо важливо в умовах розвитку дистанційного навчання та змішаних форм освітнього процесу [26].

1.3. Аналіз можливостей наявних систем.

Сучасні заклади вищої освіти активно застосовують різноманітні програмні платформи для організації та контролю лабораторних робіт. Найбільш поширеними є системи дистанційного навчання (Learning

Management Systems, LMS), серед яких Moodle, Google Classroom та комерційні корпоративні рішення. Ці платформи забезпечують базові функції управління навчальним процесом, включаючи завантаження завдань, виставлення оцінок, ведення журналів успішності та організацію комунікації між здобувачами вищої освіти та науково-педагогічними працівниками [27].

Moodle є однією з найпоширеніших систем у закладах вищої освіти України та світу. Вона надає широкі можливості для створення курсів, організації тестування та перевірки завдань. Проте, попри гнучкість у налаштуванні курсів, Moodle має обмеження щодо адаптації логіки оцінювання до специфічних вимог лабораторних робіт з програмування. Наприклад, автоматичне підрахування балів на основі складних критеріїв, інтеграція з іншими внутрішніми системами закладу або підтримка поетапного оцінювання студентських проектів реалізується лише частково і часто вимагає додаткових плагінів [28].

Google Classroom, як більш легка і доступна хмарна платформа, забезпечує простоту використання і швидке розгортання. Вона підходить для організації основних завдань та спільного доступу до матеріалів, однак не дозволяє реалізувати складну бізнес-логіку оцінювання, не підтримує інтеграцію з внутрішніми системами закладу та не забезпечує повноцінну аналітику результатів здобувачів вищої освіти [29]. Таким чином, використання цих платформ у чистому вигляді не дозволяє забезпечити повну автоматизацію процесу контролю лабораторних робіт, а також не сприяє підвищенню прозорості оцінювання та зручності для науково-педагогічних працівників.

Інші комерційні або пропрієтарні системи дистанційного навчання забезпечують розширений функціонал, включаючи інтеграцію з різними освітніми сервісами, аналітику та персоналізовані навчальні траєкторії для

здобувачів вищої освіти. Проте такі рішення часто є дорогими, складними у впровадженні та не завжди сумісні з існуючими системами закладу вищої освіти, що створює додаткові технічні та організаційні бар'єри [30].

В умовах закладів вищої освіти, де використовується кілька паралельних платформ, часто виникає проблема дублювання інформації. Наприклад, здобувачі вищої освіти виконують лабораторні роботи у Moodle, а результати мають фіксуватися у внутрішній системі управління закладу, такій як KSU24. Відсутність стандартизованого API для автоматичного обміну даними призводить до необхідності ручного перенесення інформації, що підвищує ймовірність помилок, затримує формування підсумкових балів і створює додаткове навантаження на науково-педагогічних працівників [31].

Особливу увагу слід приділити питанням безпеки та захисту персональних даних. У сучасних LMS і корпоративних системах реалізовані базові механізми авторизації та обмеження доступу, проте не завжди забезпечується комплексний контроль за збереженням результатів лабораторних робіт і можливістю їх автоматичного аналізу. Це особливо критично, коли мова йде про оцінювання здобувачів вищої освіти у великих групах, де велика кількість даних має бути захищена від несанкціонованого доступу та втрати [32].

Порівняльний аналіз існуючих освітніх платформ і систем управління навчанням показує, що жодна з наявних систем не забезпечує одночасно повного спектра функціональних можливостей, необхідних для ефективного проведення лабораторних робіт з програмування. Зокрема, відзначається, що сучасні платформи не дозволяють реалізувати повну автоматизацію оцінювання лабораторних завдань із підтримкою числових шкал та поетапного підрахунку балів. У контексті курсів програмування це створює значні труднощі, оскільки завдання часто складаються з кількох логічних

етапів, і кожен із них потребує окремої перевірки, формування проміжних балів та інтеграції результатів у підсумковий бал. Відсутність такої можливості ускладнює оперативне відстеження прогресу здобувачів вищої освіти та об'єктивну оцінку їхніх знань і практичних навичок.

Ще однією суттєвою проблемою є недостатня інтеграція з внутрішніми інформаційними системами закладу вищої освіти, зокрема KSU24, та зовнішніми платформами дистанційного навчання, такими як KSU Online на базі Moodle. Існуючі рішення не завжди забезпечують двосторонню синхронізацію даних, що призводить до дублювання інформації, затримок у оновленні результатів лабораторних робіт, а також додаткового навантаження на науково-педагогічних працівників, які змушені вручну переносити або перевіряти дані. Така ситуація знижує ефективність управління навчальним процесом та створює ризик помилок у формуванні оцінок.

Ще одне обмеження стосується надійності збереження даних та аналітики результатів. Наявні платформи не завжди забезпечують достатній рівень безпеки зберігання інформації про здобувачів вищої освіти та їхні оцінки, а також не надають гнучких інструментів для детального аналізу результатів. Це ускладнює надання оперативного зворотного зв'язку здобувачам, не дозволяє відстежувати динаміку їхнього навчання та визначати проблемні ділянки, що потребують додаткового пояснення або корекції.

Окрім цього, відзначається недостатня прозорість та стандартизація процесу оцінювання. Існуючі системи не забезпечують уніфікованих правил оцінювання для всіх науково-педагогічних працівників та здобувачів вищої освіти. Через це виникає ситуація, коли однакові завдання можуть оцінюватися по-різному залежно від викладача або групи, що знижує

об'єктивність та справедливість підсумкової оцінки. Відсутність прозорих та стандартизованих алгоритмів обробки лабораторних робіт унеможливорює формування єдиного регламентованого підходу до контролю знань у межах закладу вищої освіти.

Таким чином, проведений порівняльний аналіз демонструє, що жодна з існуючих платформ не забезпечує одночасного поєднання автоматизації оцінювання з поетапним підрахунком балів, інтеграції з внутрішніми та зовнішніми системами, надійного збереження та аналітики результатів, а також прозорості та стандартизації оцінювання, що підкреслює необхідність розробки спеціалізованої системи для підтримки лабораторних робіт із програмування.

Таким чином, наявний функціонал існуючих платформ частково покриває потреби закладів вищої освіти, але не забезпечує комплексного підходу до автоматизації лабораторних робіт. Це створює обґрунтовану необхідність розробки спеціалізованої програмної системи, яка інтегрується з існуючими цифровими середовищами закладу, зменшує адміністративне навантаження на науково-педагогічних працівників та підвищує ефективність і прозорість освітнього процесу [33].

Узагальнюючи, аналіз можливостей наявних систем дозволяє виділити ключові вимоги до нової системи: гнучкість у налаштуванні оцінювання, інтеграція з існуючими платформами, автоматичне формування підсумкових балів та збереження результатів, забезпечення безпеки та прозорості, а також адаптація до індивідуальних потреб здобувачів вищої освіти. Саме ці висновки лягли в основу постановки завдань і технічних рішень, які будуть описані у наступному розділі роботи [34].

1.4. Обмеження існуючих рішень у контексті дисциплін програмування.

Дисципліни програмування відрізняються від інших навчальних

предметів підвищеною потребою у практичних завданнях, що включають написання коду, компіляцію програм, тестування алгоритмів та інтеграцію з різними бібліотеками. В цих умовах традиційні системи дистанційного навчання, такі як Moodle або Google Classroom, хоча і забезпечують базові можливості завантаження та перевірки файлів, не дозволяють реалізувати повноцінну автоматизацію оцінювання складних практичних завдань [35]. Одним із головних обмежень є нездатність існуючих платформ підтримувати поетапну перевірку та автоматичний підрахунок балів. Наприклад, лабораторні роботи з програмування зазвичай складаються з кількох частин: написання функцій, їх тестування на контрольних даних, аналіз результатів та оформлення звіту. У більшості LMS неможливо автоматично розподілити бали між різними етапами виконання, що призводить до необхідності ручної перевірки та підвищує навантаження на науково-педагогічних працівників [36].

Іншим важливим аспектом є обмежена інтеграція з внутрішніми системами закладу вищої освіти. У багатьох закладах вищої освіти оцінки за лабораторні роботи здобувачів необхідно переносити до систем управління студентами, таких як KSU24, для формування підсумкових результатів, контролю допуску до заліку та забезпечення повного обліку академічної успішності. Цей процес часто вимагає ручного внесення даних, що значно збільшує навантаження на науково-педагогічних працівників та адміністративний персонал. Відсутність стандартних API або автоматизованих інструментів для синхронізації даних між системами призводить до дублювання роботи, уповільнення процесу оновлення оцінок та збільшує ймовірність виникнення помилок при обробці результатів. Крім того, це ускладнює оперативний доступ до актуальної інформації про навчальні досягнення здобувачів, що негативно впливає на ефективність

зворотного зв'язку та планування навчального процесу. Недостатня інтеграція також обмежує можливості аналізу даних для оцінювання тенденцій у навчанні, виявлення слабких місць у засвоєнні матеріалу та прогнозування успішності здобувачів вищої освіти, що є критично важливим для підвищення якості освітніх послуг та підтримки персоналізованих навчальних траєкторій [37].

Ще одне обмеження полягає у недостатньому контролі за якістю коду та автоматизованим тестуванням. Існуючі платформи дозволяють завантажувати файли та перевіряти текстові відповіді, але не надають інструментів для запуску автоматичних тестів, порівняння вихідних результатів із очікуваними та оцінки ефективності алгоритмів. Це особливо критично для дисциплін, де важлива не лише наявність рішення, а й його коректність, оптимальність та стиль коду [38].

Крім того, відсутня гнучкість у формуванні індивідуальних траєкторій лабораторних робіт для здобувачів вищої освіти. Кожен здобувач має різний рівень підготовки та темп освоєння матеріалу, але стандартні системи не дозволяють адаптувати завдання відповідно до індивідуальних потреб, що знижує ефективність навчання та мотивацію [39].

Безпека та захист даних також є суттєвим обмеженням. У процесі виконання лабораторних робіт здобувачі передають власні файли з кодом та звіти, що містять результати виконання завдань. Більшість LMS забезпечують базову автентифікацію та контроль доступу, але не гарантують комплексного захисту даних від несанкціонованого доступу або втрати під час інтеграції з іншими системами [40].

Важливим є й обмеження щодо аналітики та зворотного зв'язку. Для ефективного навчання необхідно не лише оцінити результат, а й надати детальний коментар, вказати помилки та рекомендації щодо їх усунення.

Стандартні платформи не дозволяють реалізувати автоматизоване формування детальної аналітики за виконані лабораторні роботи, що ускладнює підтримку здобувачів вищої освіти у процесі навчання. Узагальнюючи, обмеження наявних освітніх платформ у контексті дисциплін програмування можна виділити за кількома ключовими аспектами, які суттєво впливають на ефективність навчального процесу та оцінювання здобувачів вищої освіти.

Першим та одним із найсуттєвіших обмежень є відсутність можливості поетапного автоматизованого оцінювання складних практичних завдань. У традиційних LMS, таких як Moodle або Google Classroom, оцінювання зазвичай здійснюється на основі простих тестів або завантажених файлів із ручним переглядом. Для курсів програмування це створює значні труднощі, оскільки завдання часто складаються з кількох частин або етапів, які потребують окремої перевірки та підрахунку балів на кожному рівні. Відсутність поетапного оцінювання ускладнює контроль прогресу здобувачів вищої освіти та знижує точність формування підсумкового балу, що негативно впливає на об'єктивність оцінювання.

Другим важливим обмеженням є недостатня інтеграція з внутрішніми системами закладу вищої освіти (KSU24) та зовнішніми LMS (KSU Online/Moodle). Для ефективного управління навчальним процесом необхідно мати можливість синхронізації даних про здобувачів, групи, дисципліни та результати лабораторних робіт між різними платформами. На сьогодні наявні універсальні системи не завжди забезпечують двосторонню інтеграцію з внутрішніми базами даних закладу та зовнішніми платформами, що призводить до дублювання даних, затримок у оновленні оцінок та додаткових адміністративних витрат для науково-педагогічних працівників.

Третім аспектом є обмежені можливості автоматизованого тестування

та перевірки коду. У курсах програмування здобувачі вищої освіти виконують практичні завдання, що потребують перевірки правильності реалізації алгоритмів, стилю коду, використання структур даних та дотримання стандартів програмування. Більшість стандартних платформ не надають інструментів для автоматичного компілювання та запуску коду, перевірки тест-кейсів чи генерації детальних звітів про помилки. Це значно збільшує навантаження на науково-педагогічних працівників та знижує оперативність зворотного зв'язку для здобувачів.

Четверте обмеження стосується низької гнучкості для побудови індивідуальних навчальних траєкторій здобувачів вищої освіти. Сучасні підходи до дистанційного та змішаного навчання передбачають можливість формувати персоналізовані маршрути освоєння дисципліни, враховуючи рівень знань, інтереси та темп навчання кожного здобувача. Наявні платформи не завжди дозволяють автоматично підбирати наступні завдання або коригувати навчальний план залежно від результатів проміжних етапів, що обмежує потенціал адаптивного навчання.

Нарешті, слід зазначити недостатню безпеку та обмежену аналітику результатів. Зокрема, універсальні LMS не завжди надійно захищають персональні дані здобувачів вищої освіти та результати їхньої діяльності, що є критично важливим для відповідності законодавству про захист даних. Крім того, аналітичні інструменти обмежені стандартними статистичними звітами і не дозволяють проводити глибокий аналіз навчальних досягнень, виявляти закономірності в помилках або прогнозувати ризики невиконання завдань.

Таким чином, обмеження сучасних освітніх платформ створюють серйозні перепони для ефективного проведення лабораторних робіт з програмування, об'єктивного оцінювання та персоналізованого навчання, що

підтверджує необхідність розроблення спеціалізованої програмної системи з урахуванням потреб здобувачів вищої освіти та науково-педагогічних працівників. Врахування цих обмежень створює підґрунтя для розробки спеціалізованої системи, яка поєднуватиме автоматизацію оцінювання, інтеграцію з наявними платформами, безпечне зберігання даних та гнучкий підхід до організації лабораторних робіт.

Перш за все, таку систему необхідно проектувати з урахуванням того, що лабораторні роботи з програмування зазвичай мають складну структуру, включаючи кілька етапів виконання, тестування коду та аналізу результатів. Автоматизація оцінювання повинна передбачати не лише підрахунок підсумкового балу, але й облік проміжних результатів на кожному етапі, що дозволить науково-педагогічним працівникам отримувати детальну інформацію про прогрес здобувачів вищої освіти та своєчасно надавати рекомендації або додаткові пояснення. Такий підхід забезпечує більш точне і справедливе оцінювання, усуває суб'єктивні фактори та сприяє формуванню прозорих критеріїв оцінювання.

По-друге, інтеграція з існуючими внутрішніми системами закладу вищої освіти, такими як KSU24, та зовнішніми LMS на базі Moodle є критичною для ефективного управління навчальним процесом. Система повинна забезпечувати двосторонню синхронізацію даних про здобувачів вищої освіти, групи, дисципліни та результати лабораторних робіт. Це дозволить уникнути дублювання даних, забезпечити своєчасне оновлення оцінок та знизити навантаження на науково-педагогічних працівників, які, у протилежному випадку, змушені були б вручну переносити або перевіряти результати [42]. Інтеграція також дозволяє формувати єдину систему моніторингу прогресу здобувачів, що забезпечує більш точну аналітику та підтримку персоналізованих освітніх траєкторій.

Ще одним важливим аспектом є безпечне зберігання даних та розширена аналітика результатів. Програмна система повинна гарантувати захист персональної інформації та результатів діяльності здобувачів вищої освіти відповідно до сучасних вимог щодо конфіденційності та законодавчих норм. Крім того, система має надавати інструменти для глибокого аналізу результатів, дозволяючи виявляти закономірності у помилках, відстежувати динаміку навчання кожного здобувача та прогнозувати ризики невиконання завдань. Такий рівень аналітики сприяє оперативному зворотному зв'язку та підвищує ефективність навчального процесу.

Важливою складовою є гнучкість та адаптивність системи для побудови індивідуальних навчальних траєкторій. Сучасні підходи до дистанційного та змішаного навчання передбачають, що здобувач вищої освіти може обирати темп та послідовність освоєння матеріалу, а система автоматично підбирає завдання та контролює їх виконання. Реалізація такого підходу дозволяє підвищити мотивацію здобувачів, сприяє більш ефективному засвоєнню знань та зменшує ймовірність виникнення прогалин у навчанні. Гнучкий підхід також передбачає можливість швидкого модифікування навчальних планів та алгоритмів оцінювання залежно від вимог окремих дисциплін або груп здобувачів.

Нарешті, розроблена система повинна забезпечувати прозорість і стандартизацію процесу оцінювання для всіх науково-педагогічних працівників та здобувачів вищої освіти. Це включає уніфікацію критеріїв оцінювання, стандартизовані алгоритми підрахунку балів та прозору структуру відображення результатів. Такий підхід дозволяє створити єдині правила роботи для всіх учасників освітнього процесу, забезпечує об'єктивність оцінок та сприяє довірі здобувачів вищої освіти до системи оцінювання.

З урахуванням зазначених аспектів можна зробити висновок, що спеціалізована програмна система для підтримки лабораторних робіт з програмування має поєднувати автоматизацію оцінювання, інтеграцію з наявними платформами, безпечне зберігання даних, гнучкість побудови індивідуальних траєкторій та прозорість процесу оцінювання. Лише така система здатна забезпечити ефективну взаємодію між науково-педагогічними працівниками та здобувачами вищої освіти, підвищити якість контролю знань, оптимізувати навантаження та сприяти формуванню професійних компетентностей здобувачів.

РОЗДІЛ 2. ТЕХНОЛОГІЧНІ ЗАСАДИ ТА ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ВЗАЄМОДІЇ З KSU24 ТА KSU ONLINE.

2.1. Обґрунтування вибору технологій (React, Node.js, Express, SQLite)

Вибір технологій для розробки програмної системи підтримки лабораторних робіт є ключовим етапом проектування, оскільки він визначає продуктивність, масштабованість та зручність взаємодії для здобувачів вищої освіти та науково-педагогічних працівників. Сучасні веб-технології дозволяють створювати інтерактивні, адаптивні та безпечні освітні платформи, які можуть інтегруватися з існуючими системами, такими як KSU24 та KSU Online (Moodle) [41]. Для реалізації фронтенду обрана бібліотека React. Основні причини цього вибору:

- Компонентна архітектура. React дозволяє будувати інтерфейс із незалежних, повторно використовуваних компонентів, що значно спрощує розробку та підтримку системи [42]. Кожен елемент інтерфейсу (списки лабораторних робіт, форми завантаження, аналітичні панелі) реалізується окремим компонентом, що підвищує гнучкість системи.
- Ефективне управління станом. Використання React Hooks та Context API дозволяє централізовано управляти станом системи, синхронізувати дані між різними компонентами та забезпечує швидку реакцію інтерфейсу на дії користувача [43].
- Підтримка інтерактивності та адаптивності. React забезпечує швидке оновлення інтерфейсу без повного перезавантаження сторінки, що критично для системи лабораторних робіт, де здобувач вищої освіти часто взаємодіє з динамічними даними, такими як оцінки та коментарі науково-педагогічного працівника [44].
- Широке ком'юніті та бібліотеки. React має велику екосистему,

включаючи бібліотеки Material UI для створення сучасного та зручного користувацького інтерфейсу, що відповідає принципам UX/UI в освіті [45].

Для серверної частини системи обрана платформа Node.js разом із фреймворком Express, що забезпечує оптимальні умови для створення високопродуктивного та масштабованого програмного продукту. Node.js підтримує асинхронну обробку запитів, що дозволяє одночасно обслуговувати велику кількість користувачів без зниження швидкодії системи. Це особливо важливо для навчальних платформ, де одночасно можуть працювати десятки або сотні здобувачів вищої освіти та науково-педагогічних працівників, виконуючи лабораторні роботи, завантажуючи файли, отримуючи оцінки та формуючи підсумкові результати [46].

Вибір Express як фреймворку обумовлений його здатністю забезпечувати легку та гнучку реалізацію REST API, що є критично необхідним для організації обміну даними між фронтендом, базою даних та зовнішніми платформами, такими як KSU24 та Moodle. Express надає розвинені засоби маршрутизації, дозволяє швидко створювати RESTful сервіси, легко інтегруватися з іншими компонентами системи та впроваджувати додаткові функції без необхідності значної переробки наявного коду [47].

Ще одним важливим аргументом на користь використання Node.js та Express є можливість побудови модульної та масштабованої архітектури бекенду. Модульність дозволяє розділяти систему на незалежні компоненти, що спрощує підтримку та розширення функціоналу, зокрема інтеграцію з додатковими сервісами, такими як внутрішні бази даних закладу вищої освіти та зовнішні LMS. Масштабованість забезпечує адаптацію системи до зростання кількості користувачів та збільшення навантаження на сервер, що гарантує стабільну роботу програмного продукту у великих навчальних

зкладах

[46].

Таким чином, поєднання Node.js та Express створює міцну основу для серверної частини системи, яка поєднує високу продуктивність, легкість інтеграції з іншими компонентами та можливість подальшого розширення функціоналу без порушення існуючої архітектури.

Для зберігання даних про лабораторні роботи, користувачів та оцінки обрана база даних SQLite, яка поєднує простоту використання з високою надійністю та продуктивністю. Однією з ключових переваг цього рішення є легка інтеграція та простота налаштування. SQLite не потребує окремого сервера бази даних, що значно спрощує розгортання системи у закладі вищої освіти, особливо коли ресурси обмежені або немає можливості підтримувати складну серверну інфраструктуру [41]. Це дозволяє швидко налаштувати систему та забезпечити доступ до даних як на локальному рівні, так і у невеликих мережевих конфігураціях.

Ще однією перевагою є висока продуктивність для локальних та середніх навантажень. У випадках, коли одночасно працює обмежена кількість здобувачів вищої освіти та науково-педагогічних працівників, SQLite забезпечує швидке виконання запитів до бази даних, що гарантує оперативність обробки результатів лабораторних робіт та мінімізує затримки при взаємодії з серверною частиною [42].

SQLite також підтримує транзакції та забезпечує цілісність даних. Це критично важливо для систем, де відбувається обчислення оцінок та збереження результатів лабораторних робіт, оскільки гарантується атомарність операцій: будь-які зміни у базі даних виконуються повністю або не виконуються взагалі, що виключає ризик втрати або некоректного збереження інформації [43]. Важливим аспектом є можливість подальшого масштабування системи. У разі збільшення навантаження або розширення

функціоналу базу даних можна легко замінити на більш потужні реляційні СУБД, такі як PostgreSQL чи MySQL, при цьому мінімально змінюючи логіку бекенду та зберігаючи існуючу структуру даних. Це забезпечує гнучкість та довгострокову адаптивність розробленої системи [44].

Таким чином, використання SQLite як бази даних для зберігання інформації про лабораторні роботи та оцінки здобувачів вищої освіти створює ефективну та надійну основу для реалізації серверної частини системи, забезпечуючи простоту налаштування, продуктивність, безпеку даних та перспективи масштабування.

Комбінація React, Node.js, Express та SQLite забезпечує повну інтеграцію фронтенду та бекенду, дозволяє створювати інтуїтивно зрозумілий інтерфейс для здобувачів вищої освіти та науково-педагогічних працівників, забезпечує ефективну обробку даних та автоматизоване оцінювання лабораторних робіт. Водночас ці технології гарантують масштабованість, безпеку та можливість інтеграції з KSU24 і KSU Online (Moodle), що робить їх оптимальним вибором для розробки сучасної освітньої платформи [41]–[47].

2.2. Архітектура системи та модель інтеграції з KSU24 і KSU Online (Moodle).

Архітектура програмної системи є фундаментальною складовою, що визначає структуру взаємодії компонентів, обробку даних та забезпечення інтеграції з зовнішніми платформами [48]. Для системи підтримки лабораторних робіт обрана модульна архітектура з API-орієнтованим підходом, що дозволяє окремо розвивати фронтенд, бекенд та базу даних, а також легко підключати зовнішні сервіси, зокрема KSU24 і KSU Online (Moodle) [49]. Система підтримки лабораторних робіт з програмування

побудована на трьох основних компонентах, кожен із яких виконує чітко визначені функції та забезпечує ефективну взаємодію користувачів із системою. Першим компонентом є фронтенд, реалізований із використанням сучасного JavaScript-фреймворку React. Він відповідає за безпосередню взаємодію здобувачів вищої освіти та науково-педагогічних працівників із системою, надаючи інтуїтивний та адаптивний інтерфейс. Фронтенд забезпечує відображення списків лабораторних робіт, форм для завантаження завдань, інтерфейс для оцінювання виконаних робіт та детальні аналітичні панелі, що дозволяють користувачам отримувати зворотний зв'язок у режимі реального часу [50]. Завдяки використанню компонентного підходу React забезпечує легкість у масштабуванні інтерфейсу, можливість повторного використання компонентів та гнучкість у впровадженні нових функцій без порушення роботи вже існуючих.

Другим компонентом є бекенд, реалізований на основі Node.js у поєднанні з фреймворком Express. Цей компонент виконує роль центрального обробника запитів, забезпечуючи коректну взаємодію між фронтендом, базою даних та зовнішніми платформами, такими як KSU24 та Moodle [51]. Бекенд відповідає за виконання логіки оцінювання, включаючи автоматичний підрахунок сумарних та середніх балів, визначення допуску до підсумкового контролю, а також управління користувачами та контроль доступу до функціональних модулів системи. Крім того, бекенд реалізує механізми обробки помилок, ведення журналу дій користувачів та логування API-запитів, що забезпечує прозорість, надійність та безпеку роботи системи навіть при високому навантаженні.

Третім компонентом є база даних, вибір якої зупинився на реляційній СУБД SQLite. Ця база даних зберігає повну інформацію про користувачів, лабораторні роботи, оцінки та історію взаємодії з системою [52].

Використання SQLite дозволяє гарантувати цілісність даних, підтримку транзакцій, що критично для безпечного обчислення результатів оцінювання, а також забезпечує високу швидкість виконання запитів у системах із помірним навантаженням. Крім того, структура бази даних передбачає можливість подальшого масштабування та адаптації до інших дисциплін або розширення функціоналу, що робить систему гнучкою та стійкою до змін у навчальному процесі. Завдяки взаємодії цих трьох компонентів система забезпечує повний цикл роботи з лабораторними роботами: від подачі завдання та його виконання здобувачем до автоматизованого оцінювання, збереження результатів та інтеграції з внутрішніми й зовнішніми освітніми платформами, що значно підвищує ефективність освітнього процесу та прозорість оцінювання.

KSU24 є внутрішньою платформою закладу вищої освіти, де зберігаються дані про здобувачів вищої освіти та науково-педагогічних працівників, навчальні плани та статуси проходження дисциплін. Інтеграція здійснюється через REST API, що дозволяє:

- Синхронізація інформації про зареєстрованих здобувачів вищої освіти та їхні групи є однією з ключових функцій інтеграції з KSU24 через REST API. Ця можливість дозволяє автоматично оновлювати дані про поточних користувачів системи, забезпечуючи актуальність інформації без необхідності ручного внесення. Для програмної системи, що підтримує лабораторні роботи, це особливо важливо, оскільки дозволяє формувати правильну структуру груп, визначати склад студентів у кожній групі та коректно розподіляти завдання відповідно до навчального плану. Автоматизація цього процесу не лише економить час науково-педагогічних працівників, але й знижує ризик помилок, що можуть виникнути при ручному внесенні даних. Крім того, синхронізація забезпечує узгодженість

інформації між внутрішньою платформою KSU24 та розробленою системою, що є критичною умовою для ефективного моніторингу навчального процесу та управління академічними результатами здобувачів вищої освіти [53];

- Отримання статусів виконання лабораторних робіт через REST API дозволяє системі в режимі реального часу відстежувати прогрес здобувачів вищої освіти. Кожен зареєстрований користувач може мати різний темп виконання завдань, а автоматичний контроль їхнього виконання дозволяє науково-педагогічним працівникам своєчасно ідентифікувати проблемні місця, надавати необхідні рекомендації та організовувати додаткову підтримку. Це особливо важливо для дисциплін програмування, де ефективність навчання багато в чому залежить від практичних навичок і своєчасного засвоєння матеріалу. Доступ до актуальних статусів виконання лабораторних робіт також підвищує прозорість навчального процесу, оскільки здобувачі вищої освіти можуть самостійно відслідковувати свій прогрес і планувати роботу відповідно до встановлених термінів;

- Передача оцінок та результатів автоматизованого обчислення балів через REST API забезпечує централізоване управління результатами лабораторних робіт. Це дозволяє не тільки швидко фіксувати оцінки в системі KSU24, а й гарантує, що всі обчислення здійснюються за стандартизованими алгоритмами без людського втручання, що знижує ризик суб'єктивного оцінювання. Такий підхід забезпечує оперативність формування підсумкових результатів, полегшує підготовку допусків до підсумкового контролю і дозволяє науково-педагогічним працівникам зосередитися на якісному наданні зворотного зв'язку. Передача результатів у централізовану систему також забезпечує їхню надійну архівацію та подальшу аналітику для оцінки ефективності навчального процесу;

- Ведення журналу активності користувачів для подальшого аудиту є

важливим інструментом контролю та безпеки даних у системі. Завдяки цьому модулю можна фіксувати всі дії здобувачів вищої освіти та науково-педагогічних працівників у процесі взаємодії з платформою, включаючи завантаження робіт, зміну оцінок, доступ до матеріалів та інші ключові операції. Журнал активності дозволяє проводити ретельний аудит, визначати причини помилок або некоректних дій, а також забезпечує прозорість і підзвітність освітнього процесу. Крім того, накопичені дані можуть бути використані для статистичного аналізу поведінки користувачів, оптимізації процесів та планування навчальних заходів, що значно підвищує ефективність управління лабораторними роботами та якість освітнього процесу.

Для коректної взаємодії з KSU24 бекенд системи реалізує модулі аутентифікації, обробки помилок та логування запитів. Це дозволяє гарантувати цілісність даних та безперервність процесів оцінювання.

KSU Online базується на платформі Moodle, що забезпечує управління навчальними ресурсами, тестування та оцінювання. Інтеграція з Moodle також реалізована через REST API, що дозволяє:

- Отримання інформації про виконані завдання та результати тестування здобувачів вищої освіти через інтеграцію з KSU Online (Moodle) забезпечує автоматичне відстеження прогресу здобувачів у виконанні лабораторних робіт та інших навчальних завдань. Це дозволяє науково-педагогічним працівникам отримувати актуальні дані щодо правильності виконання завдань, кількості спроб тестування та рівня засвоєння матеріалу, що значно підвищує ефективність оцінювання та своєчасний зворотний зв'язок. Такий підхід також сприяє персоналізації навчання, оскільки система може надавати рекомендації та коригувати завдання відповідно до індивідуальних потреб кожного здобувача вищої освіти;

- Передача даних про оцінки лабораторних робіт у KSU Online дозволяє автоматично обчислювати сумарний бал кожного здобувача вищої освіти, що спрощує процес підготовки підсумкових результатів та забезпечує прозорість оцінювання. Використання автоматизованих алгоритмів для підрахунку балів знижує ймовірність помилок, які можуть виникати при ручному введенні даних, і дозволяє науково-педагогічним працівникам зосередитися на наданні якісного зворотного зв'язку та підтримці процесу навчання. Крім того, централізоване обчислення сумарного балу забезпечує уніфікований підхід до оцінювання всіх здобувачів вищої освіти, що сприяє дотриманню стандартів та академічної доброчесності;
- Синхронізація статусів допуску до заліків та контролю відвідуваності через REST API забезпечує цілісність інформації між KSU Online і внутрішньою системою закладу вищої освіти. Це дає змогу автоматично відслідковувати, чи здобувач вищої освіти виконав усі необхідні умови для допуску до підсумкового контролю, а також контролювати відвідуваність практичних та лабораторних занять. В результаті науково-педагогічні працівники отримують інструмент для своєчасного прийняття рішень щодо допуску до заліку та проведення індивідуальної роботи з тими здобувачами, які потребують додаткової підтримки, що підвищує ефективність та прозорість навчального процесі [54].

Ця інтеграція забезпечує безперервний обмін даними між системами, що підвищує прозорість оцінювання та зменшує ручну роботу науково-педагогічних працівників. Архітектура системи розроблена з урахуванням чіткого розподілу відповідальностей між компонентами та застосуванням API-орієнтованого підходу, що забезпечує ефективну взаємодію між фронтендом, бекендом та зовнішніми платформами. Фронтенд виконує роль інтерфейсу користувача та формує запити до бекенду для отримання

необхідних даних або надсилання інформації, такої як результати лабораторних робіт, завантажені файли та статуси виконання завдань. Завдяки чітко визначеному набору REST-запитів забезпечується стандартизована комунікація, що полегшує обробку інформації та мінімізує ризик помилок при передачі даних [48].

Бекенд виконує ключову роль у обробці запитів від фронтенду, здійснює взаємодію з базою даних та зовнішніми платформами, такими як KSU24 і Moodle. Він забезпечує коректне виконання бізнес-логіки, включаючи перевірку вхідних даних, обчислення підсумкових балів, формування допусків до підсумкового контролю та обробку помилок. Завдяки модульній архітектурі бекенд легко масштабується, що дозволяє додавати нові функції або підключати додаткові сервіси без суттєвих змін у структурі системи [49][50].

Зовнішні платформи, такі як KSU24 та Moodle, обробляють отримані через API дані та повертають результати, забезпечуючи інтеграцію з існуючими системами закладу вищої освіти. KSU24 надає актуальні відомості про здобувачів вищої освіти, групи, навчальні плани та статуси виконання завдань, тоді як Moodle забезпечує доступ до інформації про виконані завдання та результати тестування. Така взаємодія дозволяє синхронізувати дані між системами та підтримувати їхню актуальність у режимі реального часу, що є критично важливим для забезпечення ефективного навчального процесу [51][52].

Результати обробки та аналізу даних відображаються у фронтенді у вигляді аналітичних панелей, таблиць та детальних звітів, що дозволяє науково-педагогічним працівникам отримувати вичерпну інформацію про прогрес здобувачів вищої освіти та стан лабораторних робіт. Інформаційні панелі інтегрують дані з різних джерел, що забезпечує комплексний та

наочний контроль за виконанням завдань, а також сприяє своєчасному прийняттю рішень щодо корекції навчального процесу. Завдяки цьому архітектура системи не лише підтримує ефективність роботи користувачів, а й забезпечує прозорість та стандартизацію оцінювання [48]–[54].

Структура системи, спроектована з чітким розподілом компонентів і використанням API-орієнтованого підходу, створює умови для високого рівня масштабованості, надійності та прозорості. Масштабованість забезпечується модульною архітектурою, яка дозволяє інтегрувати нові функціональні модулі без потреби в кардинальному переписуванні існуючого коду. Це означає, що додаткові можливості, такі як нові типи лабораторних завдань, алгоритми перевірки коду, механізми автоматичного оцінювання або розширені аналітичні панелі, можуть бути впроваджені у систему поступово, з мінімальним впливом на роботу вже реалізованих функцій. Такий підхід значно знижує ризики виникнення помилок та конфліктів між компонентами, що особливо важливо у середовищі закладу вищої освіти, де одночасно працює велика кількість здобувачів вищої освіти та науково-педагогічних працівників.

Надійність системи досягається за рахунок визначення чітких меж відповідальності кожного компонента та застосування перевірених технологій для обробки запитів і збереження даних. Бекенд обробляє запити фронтенду та координує взаємодію з базою даних і зовнішніми платформами, гарантуючи цілісність інформації та безперервність процесів оцінювання. Використання транзакцій у базі даних та асинхронної обробки запитів забезпечує стабільну роботу навіть при збільшенні навантаження, що дозволяє системі залишатися працездатною у пікові моменти активності здобувачів вищої освіти.

Прозорість системи реалізується через централізовану обробку даних

та відображення результатів у фронтенді у вигляді аналітичних панелей, детальних звітів та журналів активності. Це дає можливість науково-педагогічним працівникам та адміністрації закладу вищої освіти відстежувати прогрес здобувачів, контролювати виконання завдань, перевіряти коректність оцінювання та своєчасно реагувати на проблеми. Водночас прозорість сприяє стандартизації процесу оцінювання, зменшує ризик суб'єктивності та дозволяє документально підтвердити всі виконані дії, що є важливим аспектом академічної доброчесності.

Завдяки такій структурі система підтримує гнучкість у налаштуванні алгоритмів оцінювання та логіки обчислення балів, що дозволяє легко адаптувати її до нових вимог навчальних програм або змін у державних стандартах освіти. Водночас забезпечується можливість підключення додаткових зовнішніх сервісів, наприклад, модулів для перевірки якості коду, аналітичних систем або систем управління навчальними траєкторіями здобувачів вищої освіти, без порушення стабільності роботи вже реалізованих компонентів. Така організація системи гарантує її довготривалу ефективність та зручність у використанні як для здобувачів вищої освіти, так і для науково-педагогічних працівників.

Обрана архітектура та модель інтеграції дозволяють побудувати сучасну освітню платформу, яка ефективно взаємодіє з KSU24 і KSU Online (Moodle), автоматизує процеси оцінювання лабораторних робіт та забезпечує надійну, безпечну та масштабовану роботу для здобувачів вищої освіти та науково-педагогічних працівників [48]–[54].

2.3. API-орієнтований підхід: структура та принципи взаємодії компонентів

Для забезпечення масштабованості, гнучкості та інтеграції з зовнішніми

платформами у системі підтримки лабораторних робіт застосовано API-орієнтований підхід (API-driven architecture) [55]. Цей підхід дозволяє чітко розділити функціональні блоки системи, забезпечити стандартизований обмін даними між фронтендом, бекендом та зовнішніми сервісами (KSU24 та KSU Online), а також гарантує зручність подальшого розвитку та модернізації системи. Основні принципи API-орієнтованого підходу:

- Уніфіковані точки доступу, або endpoints, є ключовим елементом архітектури системи, оскільки всі запити, що надходять від фронтенду або сторонніх систем, обробляються виключно через стандартизовані REST API, що гарантує єдину, передбачувану поведінку системи у будь-яких умовах. Такий підхід дозволяє забезпечити консистентність обробки даних, чітко визначені формати запитів та відповідей, а також стандартизовані механізми аутентифікації та авторизації, що критично для захисту конфіденційної інформації користувачів. Використання REST API спрощує масштабування системи та інтеграцію нових функціональних модулів, оскільки розробники можуть додавати нові endpoints або розширювати існуючі без ризику порушення роботи вже реалізованих компонентів. Крім того, стандартизовані точки доступу полегшують процес тестування та моніторингу, дозволяючи відстежувати коректність обробки запитів, швидко виявляти та усувати помилки, а також підтримувати прозорість взаємодії між фронтендом, бекендом та зовнішніми платформами, що підвищує надійність і стійкість системи у разі збільшення навантаження або підключення нових користувачів [56].

- Стандартизовані формати даних відіграють критично важливу роль у забезпеченні надійного та ефективного обміну інформацією між компонентами системи, оскільки всі дані передаються у форматі JSON, що є універсальним, легким для обробки та сумісним із більшістю сучасних

платформ та мов програмування. Використання JSON дозволяє фронтенду без проблем десеріалізувати отримані дані та відображати їх у користувацькому інтерфейсі, а бекенду — швидко обробляти запити та формувати відповіді у передбачуваному форматі, що значно знижує ризик помилок при передачі даних. Крім того, цей формат забезпечує повну сумісність та інтеграцію з внутрішньою системою KSU24 та зовнішньою LMS Moodle, дозволяючи автоматично передавати інформацію про користувачів, групи, результати лабораторних робіт та статуси виконання завдань. Стандартизація форматів даних також спрощує розширення функціональності системи, додавання нових модулів та інтеграцію додаткових сервісів, а також забезпечує прозорість і передбачуваність процесів обміну інформацією між усіма компонентами, що особливо важливо для безпечного та ефективного оцінювання здобувачів вищої освіти [57].

- Розділення відповідальності компонентів у системі забезпечує чітку архітектурну організацію та підвищує надійність і масштабованість усіх процесів. Фронтенд відповідає виключно за відображення інформації та збір даних від здобувачів вищої освіти та науково-педагогічних працівників, надаючи інтуїтивно зрозумілі форми для завантаження лабораторних робіт, перегляду оцінок та аналітичних панелей. Бекенд, у свою чергу, обробляє всю бізнес-логіку, здійснює розрахунок підсумкових балів, визначає допуск до заліку, управляє користувачами та взаємодіє з базою даних і зовнішніми сервісами, такими як KSU24 та Moodle, гарантуючи цілісність і коректність інформації. API виконує роль єдиного каналу зв'язку між фронтом та бекендом, стандартизуючи обмін даними, забезпечуючи передбачувану поведінку системи та спрощуючи інтеграцію нових модулів, при цьому зменшуючи ризик помилок і дублювання даних. Такий підхід дозволяє

забезпечити прозорість процесів, легкість обслуговування та подальшого розширення системи без порушення роботи вже реалізованих компонентів [58].

- Модульність та розширюваність системи реалізовані через побудову кожного функціонального блоку як окремого API-модуля, що включає, наприклад, обробку оцінок, завантаження файлів або генерацію звітів. Такий підхід забезпечує чітке розмежування відповідальності між компонентами, дозволяє незалежно оновлювати або додавати нові сервіси без впливу на роботу інших частин системи, а також спрощує тестування, налагодження та впровадження змін. Кожен модуль має власну логіку обробки даних, точки доступу та обробку помилок, що підвищує надійність і стабільність роботи системи в цілому. Завдяки цьому архітектурному рішенню можлива поступова інтеграція додаткових функцій, нових алгоритмів оцінювання або підтримки додаткових дисциплін без необхідності масштабного рефакторингу вже реалізованих компонентів, що робить систему гнучкою та адаптивною до змінних потреб закладу вищої освіти та здобувачів вищої освіти [59].

- Обробка помилок та логування в системі реалізовані через комплекс механізмів, які дозволяють API надійно реагувати на некоректні або неповні запити. Кожен запит обробляється з поверненням зрозумілих кодів помилок, що інформує користувача або інтегровану систему про природу проблеми та спосіб її усунення. Паралельно відбувається детальне логування всіх запитів, включно з інформацією про користувача, час запиту та тип виконаної операції, що забезпечує можливість відстеження дій у системі та аудиту. Крім того, API контролює автентичність та авторизацію користувачів, перевіряючи токени доступу для підтвердження прав на виконання конкретних дій. Така реалізація підвищує надійність і безпеку обміну

даними, запобігає втраті або некоректній обробці інформації, а також створює основу для ефективного моніторингу та підтримки системи в стабільному стані [60].

Архітектура API передбачає кілька рівнів взаємодії:

1. Користувацький рівень системи, відомий також як User API [61], виконує критично важливу роль у взаємодії здобувачів вищої освіти та науково-педагогічних працівників із функціоналом програмної системи. На цьому рівні реалізується безпосередня комунікація між кінцевими користувачами та серверною частиною, що забезпечує обмін даними у реальному часі та гарантує коректність і безпеку операцій. Однією з основних функцій цього рівня є авторизація та автентифікація користувачів, яка дозволяє підтвердити особу здобувача вищої освіти або науково-педагогічного працівника, забезпечуючи доступ до інформації відповідно до прав та ролей користувачів. Такий підхід не лише запобігає несанкціонованому доступу, але й дозволяє вести детальний журнал активності, що важливо для аудиту та контролю безпеки даних. Другою важливою функцією є отримання списків лабораторних робіт. За допомогою User API здобувачі вищої освіти можуть отримати актуальний перелік завдань, визначених на певний період або за конкретною дисципліною, разом із детальним описом вимог, термінами виконання та критеріями оцінювання. Для науково-педагогічних працівників цей функціонал дозволяє швидко перевіряти статуси виконання завдань усіх здобувачів, порівнювати прогрес груп та планувати необхідні консультації або додаткові активності. Такий підхід забезпечує централізоване управління інформацією, зменшуючи навантаження на адміністративний персонал та мінімізуючи ризики помилок у формуванні списків завдань. Третя функція, яка реалізується на користувацькому рівні, стосується завантаження файлів і перегляду оцінок

[61]. Здобувачі вищої освіти мають можливість безпосередньо завантажувати виконані лабораторні роботи через інтерфейс системи, що знижує ймовірність втрати матеріалів та скорочує час обробки завдань. Одночасно User API забезпечує доступ до оцінок та зворотного зв'язку, що дозволяє здобувачам оперативно відстежувати свій прогрес, коригувати помилки та планувати подальшу роботу відповідно до рекомендацій науково-педагогічних працівників. Для викладачів цей механізм автоматизує збір результатів та забезпечує централізоване відображення оцінок, дозволяючи швидко формувати підсумкові бали та здійснювати аналіз ефективності навчального процесу. Крім основних функцій, користувацький рівень підтримує додаткові операції, такі як обробка запитів на відновлення доступу, оновлення профільних даних та взаємодія з іншими модулями системи через REST API, що забезпечує гнучкість у розвитку системи та можливість її подальшого масштабування. Інтеграція цього рівня з внутрішніми та зовнішніми платформами гарантує цілісність даних і дозволяє здобувачам вищої освіти та науково-педагогічним працівникам працювати з актуальною інформацією у режимі реального часу, що підвищує ефективність навчального процесу [61].

2. Адміністративний рівень системи, позначений як Admin API, виконує ключову роль у забезпеченні повноцінного функціонування інструментів для науково-педагогічних працівників та адміністраторів освітнього середовища. На цьому рівні зосереджено функції, що стосуються управління навчальним контентом, налаштування параметрів оцінювання та контролю за результатами освітнього процесу. Він забезпечує гнучке конфігурування системи відповідно до потреб конкретної дисципліни, навчального плану або організаційних вимог. Саме Admin API дозволяє формувати структуру лабораторних робіт, контролювати їх актуальність, проводити оновлення

критеріїв, а також виконувати аналітичні операції, що є необхідними для підтримки прозорості та підвищення якості освітнього процесу. Однією з основних функцій адміністративного рівня є створення та редагування лабораторних робіт. За допомогою Admin API науково-педагогічний працівник може додавати нові завдання, визначати їх тип, обсяг, технічні вимоги, дедлайни та необхідні матеріали. Також він може вносити зміни до вже створених робіт, адаптуючи їх до оновлених програмних вимог або рівня підготовки здобувачів вищої освіти. Такий підхід дозволяє оперативно реагувати на зміни в освітньому процесі та забезпечувати актуальність лабораторних завдань. Додатково адміністративний рівень передбачає контроль за статусами робіт, можливість їх архівації або відновлення, а також керування доступом для окремих груп здобувачів. Це створює цілісну структуру роботи з навчальним контентом і мінімізує ризики дублювання або втрати інформації. Наступною важливою функцією є встановлення критеріїв оцінювання. На цьому етапі Admin API дозволяє визначити набір показників, які будуть враховуватися при перевірці лабораторних робіт, включаючи вагові коефіцієнти, поетапність виконання завдань, наявність автоматизованих тестів та максимальну кількість балів за кожен із параметрів. Також науково-педагогічний працівник може формувати індивідуальні критерії для різних груп студентів, якщо навчальним планом передбачено диференційований підхід. Завдяки цьому оцінювання стає більш прозорим, стандартизованим та узгодженим, що зменшує ризик суб'єктивності та сприяє справедливому визначенню навчальних досягнень. Третьою ключовою функцією адміністративного рівня є формування зведених звітів та аналітики. Admin API забезпечує доступ до агрегованих даних щодо успішності здобувачів вищої освіти, рівня виконання лабораторних робіт, дотримання дедлайнів та загальної динаміки навчальних

результатів. У межах цього функціоналу формуються як оперативні звіти для викладачів, так і розширені аналітичні огляди, що можуть використовуватися адміністрацією закладу вищої освіти для моніторингу ефективності освітніх програм. Аналітичні інструменти дозволяють визначати типові проблеми, виявляти здобувачів, які потребують додаткової підтримки, оцінювати навантаження на викладачів та покращувати організацію освітнього процесу в цілому. Завдяки цьому адміністративний рівень стає основою для прийняття обґрунтованих управлінських рішень і забезпечує високий рівень прозорості функціонування системи. Таким чином, адміністративний рівень системи не лише реалізує базові функції керування лабораторними роботами та оцінюванням, але й виступає стратегічним компонентом, який інтегрує аналітичні можливості, підтримує стандартизацію навчального процесу та створює умови для його подальшої оптимізації.

3. Інтеграційний рівень системи, представлений як Integration API, відіграє ключову роль у забезпеченні узгодженої та безперервної взаємодії між внутрішніми компонентами розроблюваного програмного забезпечення та зовнішніми платформами, такими як KSU24 та Moodle. Цей рівень відповідає за передачу структурованих даних, підтримку актуальності інформації та забезпечення коректності освітнього процесу, який здійснюється із залученням кількох цифрових середовищ одночасно. Завдяки інтеграційному рівню система уникає дублювання даних, зменшує ризики помилок і забезпечує єдину цифрову екосистему для навчання та оцінювання здобувачів вищої освіти. У межах Integration API реалізовано набір механізмів, які гарантують стабільність обміну інформацією, обробку помилок, захищені запити та можливість масштабування інтеграційних процесів відповідно до зростання кількості користувачів або розширення функціоналу. Саме завдяки цьому інтеграційний рівень виступає основою

для ефективної автоматизації освітнього процесу в сучасному закладі вищої освіти. Одним із ключових напрямів роботи інтеграційного рівня є синхронізація даних користувачів та груп. Integration API забезпечує отримання актуальної інформації про здобувачів вищої освіти, їхні академічні групи, статуси зарахування та приналежність до певних навчальних потоків. Ця синхронізація відбувається регулярно, щоб оновлення в KSU24 — наприклад, зміна групи, відрахування або поновлення — автоматично відображалися у внутрішній системі. Аналогічним чином, при роботі з Moodle інтеграційний рівень отримує інформацію про групи курсів, списки зареєстрованих учасників та їхнє поточне навчальне навантаження. Завдяки цьому викладачам не потрібно вручну переносити або редагувати академічні дані, а здобувачі отримують коректний доступ до потрібних лабораторних завдань. Системна синхронізація усуває ризики помилок, пов'язаних із людським фактором, та забезпечує єдність даних у межах освітнього процесу [55]–[61]. Другим важливим напрямом діяльності інтеграційного рівня є передача та отримання оцінок. Integration API забезпечує двосторонній обмін даними між внутрішньою системою оцінювання та платформами KSU24 і Moodle. Коли науково-педагогічний працівник виставляє оцінку або коли система автоматично розраховує бал за лабораторну роботу, ці результати можуть бути передані до KSU24 для подальшого формування підсумкової відомості, а також для визначення допуску здобувача до заліку або екзамену. У зворотному напрямку система може отримувати результати тестування в Moodle, які також враховуються при обчисленні загального балу за дисципліну. Така інтеграція дозволяє забезпечити прозорість та узгодженість оцінювання, уникнути необхідності дублювати оцінки вручну та мінімізувати помилки, що виникають під час трансферу даних між кількома платформами. Крім того, двосторонній обмін

забезпечує можливість побудови комплексної аналітики на основі сукупних результатів, отриманих з обох систем [55]–[61]. Третім ключовим функціональним аспектом інтеграційного рівня є контроль статусу виконання лабораторних робіт. Через Integration API система може отримувати інформацію про те, чи виконав здобувач вищої освіти лабораторну роботу, коли саме було завантажено файл, чи проходив код автоматичне тестування та які результати були отримані. У випадку роботи з Moodle інтеграційний рівень може імпортувати статуси виконання завдань, прогрес проходження модулів або підтвердження завершення тестів. Це дозволяє найбільш точно визначати реальний стан виконання навчальних завдань, оперативно інформувати здобувачів про наявні проблеми, а викладачів — про затримки або типові помилки в роботах. Такий механізм контролю сприяє своєчасному формуванню зворотного зв'язку, підвищує дисципліну виконання лабораторних завдань та робить процес оцінювання більш об'єктивним. Наявність цього функціоналу також дозволяє накопичувати дані для подальшого аудиту, аналізу успішності та вдосконалення освітнього процесу [55]–[61].

Таким чином, інтеграційний рівень виступає критично важливою складовою всієї архітектури системи, забезпечуючи стабільну взаємодію з KSU24 та Moodle, точність і повноту даних, узгодженість оцінювання та зменшення адміністративного навантаження на науково-педагогічних працівників. Він формує основу для ефективної реалізації цифрового освітнього середовища та створює передумови для його подальшого розвитку.

Принципи взаємодії компонентів:

- Принцип запит-відповідь (request-response) визначає базову модель взаємодії між фронтендом та бекендом системи і є однією з найпоширеніших

парадигм у веб-архітектурі. У цьому підході користувацький інтерфейс (фронтенд) формує чітко структурований запит до сервера, який може містити дані авторизації, параметри запиту, фільтри або дані для запису до бази. Отримавши запит, бекенд застосовує відповідну бізнес-логіку, звертається до бази даних або зовнішніх платформ, після чого повертає фронтенду структуровану відповідь у форматі JSON або іншому стандартизованому вигляді. Завдяки цьому процесу результат роботи бекенду відображається у вигляді дашбордів, таблиць, повідомлень про успішність виконання лабораторних завдань або статусів інтеграцій. Такий принцип забезпечує прозорість взаємодії, а також дозволяє стандартизувати комунікацію між компонентами системи. Це робить можливим масштабування системи, додавання нових модулів або зміни у внутрішній логіці без потреби в оновленні всього фронтенд-інтерфейсу.

- Другим фундаментальним принципом є аутентифікація та авторизація, які забезпечують безпечний доступ до ресурсів системи. Усі запити користувачів обов'язково містять токени доступу, які підтверджують їхню особу та рівень прав. Завдяки цьому система може чітко розмежовувати дії здобувачів вищої освіти, науково-педагогічних працівників та адміністраторів. Наприклад, здобувач має доступ лише до власних лабораторних робіт, оцінок і дозволених дій, тоді як викладач може переглядати та оцінювати роботи групи або потоку, а адміністратор — керувати лабораторними завданнями, критеріями та параметрами інтеграції. Використання токенів дозволяє захистити дані від несанкціонованого доступу, забезпечує контроль над виконуваними операціями та унеможливорює стороннє втручання у процеси оцінювання. Крім того, токени доступу підтримують стандарти сучасних протоколів безпеки, що робить систему стійкою до типових загроз інформаційної безпеки..

- Асинхронність обробки є надзвичайно важливим принципом для систем, у яких регулярно здійснюються ресурсоємні або тривалі операції. До таких операцій належать розрахунок сумарних оцінок за всі лабораторні роботи, формування звітності, генерація статистики та аналітичних панелей, а також передача великих масивів даних до зовнішніх систем. Асинхронна обробка дозволяє виконувати такі завдання у фоновому режимі, не блокуючи основний потік взаємодії з користувачем. Завдяки цьому фронтенд залишається чутливим та продовжує реагувати на дії здобувачів і викладачів, а користувач може продовжувати роботу, навіть якщо тривала операція ще не завершена. Такий підхід забезпечує стабільність системи та підвищує її продуктивність, що є особливо важливим у пікові періоди навантаження, наприклад наприкінці семестру або під час масового подання робіт [56].

- Принцип логування та моніторингу є ключовим для забезпечення прозорості та надійності освітнього процесу, особливо коли система працює з оцінками, персональними даними та результатами інтеграції з зовнішніми платформами. Кожен API-запит, включно зі статусом його виконання, часом обробки, користувачем, який ініціював запит, та отриманою відповіддю, фіксується у системі логування. Це дозволяє здійснювати аудит подій, виявляти помилки, аналізувати навантаження на сервер та оперативно реагувати на аномалії. Логування також допомагає у випадках конфліктних ситуацій, наприклад коли користувач стверджує, що система не зарахувала оцінку або не прийняла файл — на основі журналу подій можна чітко визначити, чи було виконано запит та як саме система на нього відреагувала. Моніторинг стану системи дозволяє контролювати доступність сервісів, виявляти збої у роботі інтеграційних модулів та забезпечувати стабільне функціонування всієї архітектури.

API-орієнтований підхід забезпечує чітку, прозору та безпечну

взаємодію між усіма компонентами системи, включно з фронтендом, бекендом та зовнішніми освітніми платформами. Такий підхід гарантує, що обмін даними відбувається у стандартизованих форматах і через формально визначені точки доступу, що суттєво знижує ризики помилок при інтеграції та забезпечує стабільність роботи навіть за умов інтенсивного навантаження або додавання нових функціональних можливостей. У межах архітектури системи підтримки лабораторних робіт API виступає центральним механізмом зв'язку, завдяки якому всі модулі взаємодіють за єдиними правилами та процедурами. Це дає можливість ізолювати логіку кожного окремого компоненту та забезпечити їх незалежний розвиток, модернізацію або заміну без погіршення чи зупинки роботи інших частин.

Завдяки використанню єдиних протоколів і структурованих форматів передачі даних, система отримує можливість забезпечувати стандартизовану взаємодію з KSU24 і Moodle. Це означає, що будь-які дані, пов'язані з користувачами, групами, лабораторними роботами чи оцінюванням, можуть передаватися уніфіковано, без потреби створювати окремі спеціалізовані механізми під кожну інтеграцію. У результаті значно спрощується розширення інтеграційного функціоналу та підтримання актуальності даних у зовнішніх освітніх середовищах.

Модульність і масштабованість системи також суттєво зростають завдяки API-центричності. Кожен функціональний компонент, що працює через API, може бути оптимізований або розширений автономно, не впливаючи на роботу інших частин. Якщо виникає потреба додати новий сервіс, тип оцінювання або модуль аналітики, це можна реалізувати шляхом підключення нового API-інтерфейсу, не змінюючи вже реалізовані механізми. Таким чином система зберігає здатність органічно масштабуватися у відповідь на зростання кількості користувачів, обсягу

даних чи складності внутрішніх процесів.

Важливим аспектом є гарантування безпечної та коректної обробки даних. API дозволяє централізовано впровадити й підтримувати стандарти автентифікації, авторизації, валідації запитів і логування. Завдяки цьому всі операції над даними, включно з переглядом, модифікацією або передаванням оцінок, здійснюються із дотриманням встановлених правил доступу й без ризику несанкціонованих маніпуляцій. Це має особливе значення, оскільки система працює з оцінками здобувачів вищої освіти, персональними даними та результатами навчальної діяльності, які підлягають підвищеним вимогам до конфіденційності та достовірності.

API-орієнтованість суттєво полегшує додавання нових функцій та інтеграцію додаткових сервісів. Потреба в удосконаленні системи або розширенні її аналітичного, методичного чи інтеграційного потенціалу може бути реалізована шляхом розроблення нових API-методів або підключення сторонніх сервісів, що також працюють через API. Це забезпечує можливість еволюційного розвитку системи без кардинальних змін у її архітектурі та без ризику порушення роботи наявних механізмів. Завдяки цьому платформа підтримки лабораторних робіт зберігає гнучкість, адаптивність і здатність ефективно відповідати на нові вимоги цифрового освітнього середовища [55]–[61].

2.4. Проектування бази даних та логіки оцінювання.

Коректне проектування бази даних є ключовим етапом розробки системи підтримки лабораторних робіт, оскільки забезпечує цілісність даних, ефективну обробку оцінок та можливість подальшого аналізу результатів здобувачів вищої освіти [62]. База даних є основним сховищем інформації про користувачів, лабораторні роботи, критерії оцінювання, завантажені

файли та результати обчислення балів. Для реалізації системи обрана реляційна СУБД SQLite, що забезпечує простоту інтеграції та надійність зберігання даних [63]. Основні таблиці та їхні призначення:

1. Users – зберігає інформацію про здобувачів вищої освіти та науково-педагогічних працівників:
 - ID користувача, ім'я, прізвище, електронна пошта;
 - роль користувача (здобувач/науково-педагогічний працівник);
 - прив'язка до групи та дисципліни [64].
2. LabWorks – містить інформацію про лабораторні роботи:
 - ID роботи, назва, опис, дата проведення;
 - максимальний бал, критерії оцінювання, зв'язок із дисципліною [65].
3. Submissions – таблиця для зберігання файлів, завантажених здобувачами вищої освіти:
 - ID завантаження, ID користувача, ID лабораторної роботи;
 - дата та час завантаження, статус перевірки, посилання на файл [66].
4. Grades – таблиця оцінок:
 - ID оцінки, ID завантаження, оцінка (0–5);
 - коментар науково-педагогічного працівника, дата внесення;
 - автоматично обчислений сумарний та середній бал для здобувача [67].
5. AuditLog – журнал дій користувачів:
 - ID запису, ID користувача, дія, дата та час;
 - використовується для контролю достовірності даних та аудиту [68].

Ця структура дозволяє забезпечити цілісність даних, відстежувати історію змін та надавати аналітичні звіти для науково-педагогічних працівників і адміністрації закладу вищої освіти. Система підтримує поетапне оцінювання лабораторних робіт із автоматичним обчисленням сумарного та середнього балу [69]. Основні принципи логіки:

1. Бальна шкала 0–5 – кожна лабораторна робота оцінюється за п'ятибальною шкалою, що відповідає стандартам закладу вищої освіти.
2. Автоматичний розрахунок сумарного балу – при введенні оцінок система автоматично обчислює сумарний бал за всі виконані лабораторні роботи для кожного здобувача вищої освіти.
3. Середній бал та допуск до заліку – на основі сумарного балу та встановлених критеріїв система визначає, чи здобувач вищої освіти допущений до заліку.
4. Підтримка коментарів та рецензій – науково-педагогічний працівник може додавати текстові коментарі до оцінок, що відображаються здобувачу вищої освіти у фронтенді.
5. Інтеграція з зовнішніми платформами – оцінки та результати синхронізуються з KSU24 і Moodle через API, що дозволяє уникнути дублювання даних та забезпечує актуальність інформації [62]–[69]. Проектування бази даних та логіки оцінювання є фундаментальною складовою системи, що забезпечує надійне зберігання та обробку даних. Коректна структура таблиць, продумані індекси та правильно спроектовані зв'язки між сутностями (здобувачі вищої освіти, науково-педагогічні працівники, лабораторні роботи, завантаження, результати оцінювання, журнали подій тощо) дозволяють гарантувати цілісність і цілеспрямованість інформації. Прикладом таких рішень є використання транзакцій при операціях запису, механізмів довільної блокування для уникнення конкурентних конфліктів і контрольних перевірок цілісності (foreign key constraints, unique constraints). Також важливим є збереження метаданих (версії завдань, історія змін критеріїв, відмітки часу та ідентифікатори обробки), що дає можливість відтворювати стан системи на будь-який момент часу та проводити аудит. У поєднанні з належною політикою бекапів

і процедурою відновлення, така архітектура забезпечує стійкість до апаратних і програмних збоїв, мінімізує ризик втрати даних і дозволяє адміністраторам оперативно реагувати на інциденти.

Коректність автоматичного підрахунку балів досягається за рахунок чітко формалізованої логіки оцінювання, що реалізована на рівні бізнес-логіки бекенду і підтверджується наборами модульних і інтеграційних тестів. Визначення правил підрахунку — це не лише проста сума проміжних оцінок: система повинна підтримувати вагові коефіцієнти для різних етапів, правила округлення, політики врахування повторних спроб, обмеження за часом і умови допуску до підсумкового контролю. Логіка має бути виражена у вигляді детально документованих процедур або мікросервісів, що повертають однозначні результати за заданими вхідними даними. Для підвищення довіри до автоматизованого підрахунку доцільно впровадити механізми перевірки — наприклад, подвійний обрахунок у тестовому режимі, логування проміжних кроків обчислення, надання можливості ручного перегляду або коригування з фіксацією причин змін. Це дозволяє не лише забезпечити математичну коректність, а й зробити процес підрахунку прозорим і відтворюваним.

Прозорість та зручність взаємодії для здобувачів вищої освіти та науково-педагогічних працівників забезпечуються шляхом продуманого представлення даних і механізмів доступу. Структура бази даних має відображати ту модель інформації, яка зручна для користувачів: зрозумілі записи про кожен лабораторну роботу, деталі поетапних оцінок, коментарі рецензента, часові позначки, історія версій і можливість завантажити оригінали файлів. На староні фронтенду це реалізується через дашборди, фільтри, можливість перегляду детальних звітів і експорту даних у стандартні формати. Для науково-педагогічних працівників важливі

інструменти групового перегляду, масового оцінювання, шаблони коментарів і механізми швидкого корегування помилок з фіксацією аудиту. Для здобувачів – миттєвий доступ до результатів, пояснень та історії змін. Такий підхід знижує когнітивне навантаження, підвищує довіру до системи і сприяє оперативному навчальному процесу.

Швидка інтеграція з зовнішніми платформами досягається завдяки уніфікованій моделі даних і стандартизованим інтерфейсам обміну. При проєктуванні бази даних враховуються поля для зовнішніх ідентифікаторів (наприклад, `ksu24_id`, `moodle_user_id`, `external_assignment_id`), що дозволяє здійснювати однозначне співставлення сутностей під час синхронізації. Архітектура повинна підтримувати як *push*-механізми (пересилка змін у зовнішні системи), так і *pull*-механізми (періодичне опитування зовнішніх API), а також механізми виправлення конфліктів (*conflict resolution*), якщо джерела мають різні версії даних. Наявність проміжного шару трансформації даних (ETL або адаптерів API) полегшує перетворення форматів, валідацію та логування обміну, що суттєво скорочує час інтеграції й знижує ризик помилок при підключенні нових платформ.

Можливість подальшого масштабування та адаптації системи до інших дисциплін закладається вже на етапі логічного проєктування. Хороша реляційна модель із чіткою нормалізацією та помірною денормалізацією там, де це виправдано з точки зору продуктивності, дозволяє розширювати набір сутностей (наприклад, нові типи завдань: проекти, практичні лабораторні, контрольні роботи) без зміни базових механізмів обліку. Крім того, проєктування із застосуванням шарів абстракції доступу до даних (DAO/Repository pattern) і визначенням версій схеми бази даних (migrations, versioning) дають змогу плавно мігрувати на більш потужні СУБД або розподілені сховища при зростанні навантаження. Адаптивність також

передбачає параметризованість правил оцінювання (конфігуровані політики) та інтернаціоналізацію, що робить систему придатною для широкого кола дисциплін і різних навчальних середовищ [62]–[69].

2.5. Забезпечення безпеки, масштабованості та коректної взаємодії зі сторонніми сервісами

Ефективна робота системи підтримки лабораторних робіт в закладі вищої освіти потребує надійної організації безпеки, масштабованої архітектури та стабільної взаємодії з зовнішніми платформами, такими як KSU24 та KSU Online (Moodle) [70]. Безпека системи:

- Аутентифікація та авторизація користувачів. Система використовує токени доступу та JWT (JSON Web Tokens) для підтвердження прав користувача. Кожен здобувач вищої освіти та науково-педагогічний працівник отримує унікальний токен після успішної авторизації, що дозволяє обмежити доступ до даних та функціоналу відповідно до ролі [71].
- Шифрування даних. Для збереження конфіденційних даних (логіни, паролі, особисті дані здобувачів вищої освіти) застосовується хешування паролів із використанням bcrypt, а обмін даними між фронтендом, бекендом та зовнішніми сервісами відбувається через HTTPS, що гарантує захист від перехоплення інформації [72].
- Валідація та обробка помилок. Всі вхідні дані перевіряються на рівні бекенду для запобігання SQL-ін'єкціям та іншим типам атак. Система повертає стандартизовані коди помилок та повідомлення для безпечного оповіщення користувача [73].
- Аудит дій користувачів. Ведеться журнал усіх дій здобувачів вищої освіти та науково-педагогічних працівників, що дозволяє відстежувати потенційні порушення та забезпечує прозорість процесу оцінювання [74].

Масштабованість:

- Модульна архітектура. Розділення системи на фронтенд, бекенд та базу даних дозволяє незалежно розширювати компоненти, додавати нові функції та підключати додаткові модулі без порушення роботи вже існуючих сервісів [75].
 - API-орієнтований підхід. Завдяки стандартизованим REST API нові сервіси можуть бути інтегровані до системи з мінімальними змінами в існуючому коді. Це забезпечує легке масштабування системи для одночасної роботи великої кількості здобувачів вищої освіти [76].
 - Можливість інтеграції з іншими базами даних. Використання SQLite як стартового рішення дозволяє у майбутньому перейти на більш потужні СУБД (PostgreSQL, MySQL) у разі зростання навантаження або потреби у масштабованості [77].
- Коректна взаємодія зі сторонніми сервісами:
- Стандартизовані формати обміну. Взаємодія з KSU24 і Moodle здійснюється через REST API із форматом даних JSON, що забезпечує коректну передачу інформації про користувачів, групи, лабораторні роботи та оцінки [70].
 - Синхронізація даних. Система реалізує механізми перевірки актуальності інформації: при будь-якій зміні даних у KSU24 чи Moodle автоматично оновлюються записи у внутрішній базі даних, що гарантує узгодженість даних та уникнення дублювання [71].
 - Обробка помилок при інтеграції. Для забезпечення безперервної роботи система включає механізми повторних спроб передачі даних, логування помилок інтеграції та оповіщення адміністратора у разі критичних збоїв [72].
 - Контроль доступу на стороні сервісів. Всі запити до зовнішніх платформ проходять аутентифікацію та авторизацію на рівні API, що

забезпечує доступ тільки до дозволених даних та запобігає несанкціонованим операціям [73].

Реалізація комплексних механізмів безпеки, масштабованості та коректної інтеграції є критично важливим етапом розробки системи підтримки лабораторних робіт, оскільки вона визначає стабільність, надійність і довіру користувачів до платформи. Захист конфіденційних даних здобувачів вищої освіти та науково-педагогічних працівників досягається за рахунок багаторівневої системи автентифікації та авторизації. Кожен користувач отримує унікальні токени доступу, які підтверджують його права та роль у системі, а всі запити перевіряються на відповідність цим правам. Крім того, впроваджуються механізми шифрування даних як при збереженні, так і під час передачі, що гарантує конфіденційність і захист від несанкціонованого доступу, включно з протоколами HTTPS та TLS. Логування подій користувачів та системних операцій дозволяє відстежувати дії, проводити аудит і оперативно реагувати на потенційні порушення безпеки [70].

Ефективність роботи системи при зростанні навантаження забезпечується застосуванням масштабованої архітектури. Використання модульного підходу, асинхронної обробки запитів на рівні бекенду та оптимізованих запитів до бази даних дозволяє одночасно обслуговувати великий потік здобувачів вищої освіти та науково-педагогічних працівників без зниження швидкодії або якості обробки даних. Додатково, застосування кешування та балансування навантаження дозволяє уникнути перевантаження окремих компонентів системи та забезпечує швидкий доступ до найбільш затребуваних ресурсів. Така структура дозволяє системі адаптуватися до росту кількості користувачів або збільшення обсягу даних без значних змін у коді або базі даних [71][72].

Надійна взаємодія з зовнішніми платформами KSU24 і Moodle реалізується через стандартизовані API та механізми синхронізації даних. Система здатна коректно передавати оцінки та результати виконання лабораторних робіт, отримувати статуси допуску до заліків, перевіряти відповідність групових списків та оновлювати інформацію про користувачів у реальному часі. Для забезпечення цілісності даних передбачені механізми підтвердження успішної передачі (acknowledgment), контроль версій та відновлення у разі конфліктів. Такий підхід гарантує, що будь-які зміни у зовнішніх системах або внутрішніх даних будуть синхронізовані без втрати інформації і з можливістю аудиту [73][74].

Збереження узгодженості даних та прозорості оцінювання досягається завдяки єдиному джерелу правди (single source of truth) та централізованому управлінню логікою оцінювання. Всі операції підрахунку балів, обчислення підсумкових результатів та формування допусків відбуваються у визначеному порядку, із збереженням історії змін та можливістю відновлення попередніх станів. Це забезпечує прозорість для здобувачів вищої освіти та науково-педагогічних працівників, дозволяючи отримувати детальні звіти про кожен етап оцінювання, порівнювати очікувані та фактичні результати та своєчасно реагувати на помилки або непередбачувані ситуації. Механізми версіонування даних та процедур забезпечують відтворюваність результатів і дозволяють здійснювати контроль за дотриманням політик оцінювання [75][76][77].

Таким чином, комплексне впровадження заходів безпеки, масштабованості та інтеграції забезпечує надійну, стабільну та адаптивну роботу системи підтримки лабораторних робіт. Це дозволяє ефективно використовувати систему у закладах вищої освіти, забезпечує довіру

користувачів, підтримує високі стандарти захисту інформації та створює підґрунтя для подальшого розвитку та розширення функціоналу. [70]–[77]

Ці характеристики створюють основу для стійкої, безпечної та довгостроково підтримуваної освітньої платформи, готової до інтеграції нових функцій та розширення.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ТА ОЦІНЮВАННЯ ЇЇ ЕФЕКТИВНОСТІ

3.1. Реалізація фронтенд-модуля.

Фронтенд системи реалізовано з використанням сучасного JavaScript-фреймворку React, що дозволяє створювати динамічні та інтерактивні веб-додатки з високим рівнем продуктивності та гнучкістю у налаштуванні компонентів. Основним завданням фронтенду є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для здобувачів вищої освіти та науково-педагогічних працівників, що дозволяє ефективно взаємодіяти з системою без необхідності проходження тривалого навчання. Для побудови візуальної частини використано бібліотеку Material UI, яка надає готові компоненти у відповідності з принципами матеріального дизайну, забезпечуючи естетичність, узгодженість стилю та швидке створення адаптивних елементів інтерфейсу. Завдяки цьому, користувач отримує стабільний та зрозумілий досвід взаємодії з платформою незалежно від рівня своєї цифрової підготовки.

Авторизація та панель користувача є ключовими функціональними блоками фронтенду. Модуль авторизації забезпечує безпечний вхід у систему, використовуючи токени доступу, що гарантують відповідність прав користувача його ролі — здобувач вищої освіти або науково-педагогічний працівник. Після успішного входу користувач отримує доступ до персоналізованої панелі, на якій відображаються доступні лабораторні роботи, стан їх виконання та попередні результати. Панель користувача реалізована з урахуванням принципів зручності навігації та швидкого доступу до найбільш затребуваних функцій, що значно зменшує час на пошук необхідної інформації та підвищує ефективність використання

системи.

Сторінки лабораторних робіт реалізують функціонал детального відображення завдань та критеріїв оцінювання. Кожна лабораторна робота супроводжується описом завдання, інформацією про очікувані результати, формалізованими критеріями оцінювання та прикладами виконання, що дозволяє здобувачам вищої освіти більш чітко розуміти вимоги до роботи. Інтерактивні елементи на сторінках дозволяють завантажувати файли з результатами лабораторних робіт, переглядати коментарі науково-педагогічних працівників та відстежувати прогрес виконання завдань. Така структура сприяє підвищенню мотивації здобувачів вищої освіти та дозволяє систематизувати процес навчання, роблячи його більш прозорим та керованим.

Система оцінювання та відображення результатів є окремим блоком, що забезпечує зручне відображення підсумкових показників лабораторних робіт. Користувач має можливість переглядати сумарні бали, середній результат по дисципліні та коментарі науково-педагогічного працівника, що дозволяє отримати повну картину успішності. Інтерфейс реалізований таким чином, щоб усі показники були доступні у наочній формі, включно з графічними панелями та таблицями, що підвищує зрозумілість та швидкість аналізу даних.

Адаптивність і зручність користування є невід'ємною складовою фронтенду. Всі компоненти інтерфейсу автоматично підлаштовуються під різні роздільні здатності екранів та типи пристроїв, що критично важливо для здобувачів вищої освіти, які можуть використовувати планшети, ноутбуки або мобільні телефони для виконання лабораторних робіт у дистанційному режимі. Завдяки адаптивності забезпечується однаковий рівень функціональності та комфорту користування незалежно від платформи

доступу.

Інтерація фронтенду з бекендом здійснюється через REST API, що забезпечує обмін даними у режимі реального часу. Завдяки цьому система миттєво оновлює інформацію про статуси лабораторних робіт, оцінки та коментарі, синхронізуючи їх із зовнішніми платформами KSU24 і Moodle. Такий підхід дозволяє забезпечити цілісність даних, прозорість оцінювання та оперативний зворотний зв'язок між здобувачами вищої освіти та науково-педагогічними працівниками, підвищуючи ефективність освітнього процесу та зменшуючи ймовірність помилок або втрати інформації.

3.2. Реалізація бекенд-модуля та серверної логіки

Бекенд системи побудовано на основі Node.js із використанням фреймворку Express, що забезпечує високу продуктивність та гнучкість при розробці серверної частини. Основним завданням бекенду є обробка запитів від фронтенду, взаємодія з базою даних та зовнішніми платформами, а також забезпечення безпечної та коректної логіки оцінювання лабораторних робіт. Компоненти бекенду спроектовані з урахуванням принципів модульності та масштабованості, що дозволяє легко додавати нові функції, інтегрувати додаткові сервіси та адаптувати систему до змін у вимогах освітнього процесу.

Обробка REST-запитів є ключовим компонентом бекенду. Вона реалізована через чітко структуровані маршрути, які забезпечують авторизацію та автентифікацію користувачів, отримання списків лабораторних робіт, завантаження та збереження файлів із результатами, а також запис та отримання оцінок. Кожен маршрут спроектовано таким чином, щоб забезпечити безпеку переданих даних, перевірку коректності запитів та швидку обробку навіть при великій кількості одночасних користувачів. Завдяки використанню асинхронних операцій Node.js система

здатна ефективно обробляти численні запити, що важливо для підтримки одночасної роботи здобувачів вищої освіти та науково-педагогічних працівників.

Бізнес-логіка оцінювання є окремим функціональним блоком, який автоматизує процес підрахунку сумарного та середнього балів за лабораторні роботи, а також визначає допуск до заліку. Цей модуль забезпечує прозорість та об'єктивність оцінювання, виключаючи можливість помилок, що виникають при ручному підрахунку. Крім того, бізнес-логіка реалізує контроль за відповідністю введених даних встановленим критеріям оцінювання, що дозволяє підтримувати єдину систему стандартів для всіх здобувачів вищої освіти та науково-педагогічних працівників.

Обробка помилок та логування є невід'ємною складовою бекенду, що гарантує стабільність і надійність роботи системи. У разі некоректного введення даних або виникнення системних помилок модуль забезпечує їх фіксацію, відображає коректні повідомлення користувачу та зберігає інформацію про події у журналі дій для подальшого аудиту. Це дозволяє не лише відслідковувати активність користувачів, а й аналізувати проблемні ситуації, забезпечуючи постійне вдосконалення роботи системи та підтримку високого рівня безпеки.

Масштабованість та розширюваність бекенду забезпечується модульною структурою коду. Кожна функція або логічний блок ізольовані, що дозволяє додавати нові маршрути, алгоритми оцінювання або інтеграції з додатковими сервісами без порушення існуючої логіки. Така архітектура забезпечує гнучкість у розвитку системи, дозволяючи оперативно реагувати на зміни у навчальному процесі та вимогах закладу вищої освіти.

Бекенд взаємодіє з базою даних SQLite для зберігання інформації про лабораторні роботи, користувачів та результати оцінювання, а також з

зовнішніми платформами KSU24 та Moodle через стандартизовані REST API. Такий підхід забезпечує цілісність даних, своєчасне оновлення інформації та узгодженість результатів оцінювання, створюючи надійну та прозору систему підтримки лабораторних робіт у закладі вищої освіти.

3.3. Інтеграція з KSU24 та KSU Online через REST API

Інтеграція системи підтримки лабораторних робіт передбачає повноцінну синхронізацію даних користувачів, груп та оцінок між внутрішніми та зовнішніми освітніми платформами, що забезпечує цілісність інформації та ефективність процесу оцінювання. Першим ключовим компонентом інтеграції є KSU24, внутрішня інформаційна система закладу вищої освіти. Через REST API система отримує актуальні дані про здобувачів вищої освіти, їхні навчальні групи та дисципліни, а також передає результати виконаних лабораторних робіт, включаючи сумарні та середні бали, коментарі науково-педагогічних працівників та інформацію про допуск до підсумкового контролю. Така взаємодія дозволяє автоматизувати передачу результатів, зменшити ризик помилок при ручному введенні даних та забезпечує актуальність інформації для всіх зацікавлених сторін.

Другим компонентом інтеграції є KSU Online, що базується на платформі Moodle. Через цю платформу система отримує інформацію про завдання та результати тестування здобувачів вищої освіти, що дозволяє відстежувати прогрес виконання лабораторних робіт та оцінювати знання здобувачів у реальному часі. Система також передає оцінки та коментарі науково-педагогічного працівника, що забезпечує автоматичне обчислення сумарного балу та формування підсумкових результатів. Крім того, інтеграція з Moodle дозволяє синхронізувати статуси допуску до заліків та контроль відвідуваності, що підвищує прозорість та ефективність

навчального

процесу.

Механізм обробки помилок інтеграції є критично важливим для забезпечення стабільності системи. У разі збоїв або некоректної передачі даних передбачені автоматичні повторні спроби відправлення запитів, логування всіх подій та повідомлення адміністратора про виниклі проблеми. Це дозволяє оперативно реагувати на неполадки, підтримувати цілісність даних та гарантувати безперервність процесів оцінювання.

Для забезпечення коректного обміну інформацією використовується формат JSON, що гарантує сумісність між різними платформами та дозволяє передавати структуровані дані у форматі, зручному для обробки на фронтенді та бекенді системи. Завдяки цьому всі елементи даних — від списку лабораторних робіт до оцінок та коментарів — відображаються коректно на обох платформах і забезпечують уніфікований процес оцінювання, що виключає дублювання даних та підвищує прозорість і точність результатів.

Загалом інтеграція з KSU24 та Moodle створює єдину екосистему для обробки лабораторних робіт, забезпечуючи ефективний обмін даними, надійне зберігання інформації та оперативний зворотний зв'язок для здобувачів вищої освіти та науково-педагогічних працівників.

3.4. Тестування функціоналу, інтерфейсу та взаємодії компонентів

Тестування програмної системи проводилося у три послідовні етапи, що дозволило комплексно оцінити її функціональні можливості, зручність користування та коректність взаємодії компонентів. Перший етап — функціональне тестування — передбачав детальну перевірку всіх основних функцій системи. Було протестовано процес авторизації та автентифікації здобувачів вищої освіти та науково-педагогічних працівників, завантаження та обробку файлів лабораторних робіт, автоматичне обчислення сумарних та

середніх балів, визначення допуску до підсумкового контролю, а також синхронізацію даних із внутрішньою системою KSU24 та зовнішньою платформою KSU Online/Moodle. Цей етап дозволив переконатися, що всі алгоритми обробки даних працюють коректно, а взаємодія між різними модулями системи забезпечує цілісність та точність результатів.

Другий етап — UI/UX тестування — був спрямований на оцінку зручності користування інтерфейсом та адаптивності відображення інформації на різних пристроях, включно з десктопними комп'ютерами, планшетами та смартфонами. Під час цього тестування було перевірено логіку навігації, швидкість відображення даних, інтуїтивність елементів управління та коректність відображення критеріїв оцінювання, результатів лабораторних робіт і коментарів науково-педагогічних працівників. Особлива увага приділялася відповідності дизайну Material UI стандартам доступності та комфортності користування, що є критично важливим у дистанційних та змішаних формах навчання. За результатами цього етапу було виявлено незначні недоліки у відображенні коментарів на мобільних пристроях, які були оперативно усунені шляхом доопрацювання стилів та компонентів бібліотеки Material UI.

Третій етап — тестування взаємодії компонентів — передбачав комплексну перевірку коректності обміну даними між фронтендом, бекендом, базою даних та зовнішніми платформами. Було перевірено, як система обробляє запити від користувачів, як бекенд коректно зберігає дані у базі SQLite та передає їх у KSU24 і Moodle, а також як відбувається отримання актуальної інформації про виконані завдання та статуси лабораторних робіт. Особлива увага приділялася обробці помилок інтеграції та логуванню запитів для забезпечення прозорості та надійності роботи системи. Результати цього етапу підтвердили стабільність системи,

відсутність критичних помилок у взаємодії компонентів та правильність автоматичного обчислення підсумкових балів.

Загалом, проведене тестування дозволило переконатися у високій функціональності, зручності та надійності розробленої системи підтримки лабораторних робіт, а також підтвердило доцільність її використання у закладах вищої освіти для автоматизації процесів оцінювання та інтеграції з наявними освітніми платформами.

3.5. Аналіз результатів, виявлені недоліки та перспективи розвитку системи*

Проведений аналіз функціональних можливостей та ефективності розробленої системи показав, що вона успішно виконує ключові завдання, для яких була створена. По-перше, система забезпечує коректне автоматичне обчислення балів за лабораторні роботи здобувачів вищої освіти та формує допуск до підсумкового контролю відповідно до визначених критеріїв оцінювання. Це дозволяє науково-педагогічним працівникам уникнути рутинного ручного підрахунку балів і мінімізує ризик помилок, забезпечуючи точність та об'єктивність оцінювання. По-друге, реалізовані механізми взаємодії користувачів із системою створюють зручний інтерфейс для здобувачів вищої освіти та науково-педагогічних працівників, що дозволяє ефективно виконувати та перевіряти лабораторні роботи, переглядати оцінки та коментарі, а також отримувати зворотний зв'язок у режимі реального часу. По-третє, інтеграція з внутрішньою системою KSU24 та зовнішньою платформою Moodle відбувається стабільно, без втрати даних, що забезпечує цілісність інформації та уніфікований підхід до формування результатів навчання. Нарешті, модульна та масштабована структура системи дозволяє додавати нові функції, підключати додаткові дисципліни та адаптувати систему до зростання кількості користувачів без порушення

роботи вже реалізованих компонентів, що є ключовим для підтримки ефективного освітнього процесу у закладах вищої освіти.

Водночас під час аналізу та апробації системи були виявлені певні недоліки, які потребують додаткової уваги у подальшому розвитку. Зокрема, необхідне додаткове тестування роботи системи при значному одночасному навантаженні, коли в системі одночасно працює велика кількість здобувачів вищої освіти та науково-педагогічних працівників. Це дозволить переконатися в стабільності роботи сервісів та відсутності затримок при обробці запитів. Також було визначено потребу в оптимізації запитів до бази даних, особливо у випадках зростання обсягу збережених даних та збільшення інтенсивності роботи з ними. Оптимізація дозволить забезпечити швидкий доступ до результатів та зберегти ефективність автоматичного підрахунку балів навіть при високих навантаженнях.

Перспективи розвитку системи передбачають впровадження аналітичних модулів, які дозволять здійснювати детальний аналіз навчальної активності здобувачів вищої освіти, виявляти прогалини у виконанні лабораторних завдань та надавати рекомендації для покращення навчального процесу. Додатковою перспективою є інтеграція з іншими дистанційними платформами, що забезпечить більш гнучке та комплексне середовище для організації навчання. Крім того, планується покращення адаптивності інтерфейсу для мобільних пристроїв, що дозволить здобувачам вищої освіти та науково-педагогічним працівникам комфортно працювати з системою на будь-яких типах пристроїв та у різних умовах, включно з дистанційним навчанням або навчанням у змішаному форматі. Загалом, проведений аналіз підтверджує ефективність системи та окреслює шляхи її подальшого вдосконалення, що сприятиме підвищенню якості освітнього процесу та прозорості оцінювання в закладах вищої освіти.

ВИСНОВКИ

Актуальність проведеного дослідження зумовлена необхідністю підвищення ефективності організації та оцінювання лабораторних робіт з програмування в умовах інтенсивного розвитку дистанційних та змішаних форм навчання. Розроблена програмна система дозволяє забезпечити прозорість оцінювання, автоматизацію обробки лабораторних завдань та підрахунок підсумкових балів, що є критично важливим для закладів вищої освіти, де формується професійна компетентність здобувачів вищої освіти ІТ-спеціальностей.

В ході дослідження було здійснено детальний аналіз існуючих програмних рішень для організації та контролю лабораторних робіт, включаючи універсальні освітні платформи Moodle та Google Classroom. На основі цього аналізу були визначені обмеження цих систем у контексті дисциплін з програмування, що зумовило потребу у створенні спеціалізованої платформи. Визначено функціональні та нефункціональні вимоги до програмної системи, зокрема автоматизацію оцінювання, інтеграцію з платформами KSU24 та KSU Online, забезпечення масштабованості та безпеки даних.

Було розроблено архітектуру системи, яка забезпечує модульність та можливість подальшого розширення функціоналу без порушення існуючої логіки. У рамках практичної реалізації виконано розробку фронтенд-модуля за допомогою React та Material UI, що дозволяє створити зручний і інтуїтивно зрозумілий інтерфейс для здобувачів вищої освіти та науково-педагогічних працівників. Бекенд-модуль реалізовано на основі Node.js та Express, включно з логікою оцінювання лабораторних робіт, обробкою запитів та взаємодією з

базою даних SQLite. Створено структуровану базу даних для зберігання результатів лабораторних робіт, що забезпечує їхню надійну обробку та подальший аналіз.

Важливою складовою реалізації системи стала інтеграція компонентів через REST API, що дозволяє синхронізувати дані з платформами KSU24 та KSU Online, забезпечуючи безперебійну взаємодію між фронтендом і бекендом. Проведене тестування підтвердило функціональність системи, коректність роботи інтерфейсу, стабільність інтеграції з зовнішніми платформами та ефективність автоматизованого оцінювання. Результати тестування дозволили усунути незначні недоліки у відображенні інформації на мобільних пристроях та оптимізувати обробку даних.

Результати дослідження були апробовані під час внутрішнього тестування на кафедрі комп'ютерних наук та програмної інженерії, що дозволило оцінити ефективність реалізованих рішень і отримати практичні рекомендації щодо подальшого вдосконалення функціоналу системи. У підсумку виконані завдання дослідження підтвердили доцільність обраного підходу та можливість практичного застосування розробленої програмної системи у закладах вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Xie, Chengxia & Li, Chen & Mo, Fan & Wang, Tingli. (2025). Exploring the Integration of Artificial Intelligence in Clinical Laboratory Diagnostics Education: Opportunities and Challenges. Pacific International Journal. 8. 6-11. 10.55014/pij.v8i5.869.
2. Mufron, Ali & Vann, Dara. (2025). Integrating Service Learning in Higher Education: Developing Civic Responsibility and Practical Skills through Real-World Engagement. Sciencetechno: Journal of Science and Technology. 3. 328-340. 10.70177/sciencetechno.v3i3.1741.
3. Feng, Tingting & Huang, Lijian & Peng, Xiang & Qiao, Tianhui & Wu, Xue. (2025). An integrative review of cognitive workload assessment for safety management. BMC Nursing. 24. 10.1186/s12912-025-03987-w.
4. Vicente, Evelyn & Paglinawan, James. (2025). School Infrastructure Availability and Emotional Flexibility: Their Roles in Mitigating Digital Engagement Overload Among Basic Education Teachers. Volume 8. Page: 11-24. 10.37502/IJSMR.2025.81102.
5. Patra, Greece. (2006). 3rd International Conference of Open and Distance Learning (ICODL 2005) held in Patra, Greece. The Turkish Online Journal of Distance Education. 7
6. Goliath, Steve & Tsibolane, Pitso & Snyman, Dirk. (2025). Exploring the Cybersecurity-Resilience Gap: An Analysis of Student Attitudes and Behaviors in Higher Education. 10.1007/978-3-032-09660-9_19.
7. Mamonov, Kostiantyn & Afanasiev, Aleksandr & Vasyl, Goi & Viatkin, R.. (2025). STANDARDIZATION OF ASSESSMENT OF THE LEVEL OF LAND USE OF REGIONS: IMPLEMENTATION PRACTICE. Наука і техніка сьогодні. 10. 1286-1295.

8. Analytics, Learning. (2018). Companion Proceedings of the 8th International Conference on Learning Analytics & Knowledge (LAK'18).
9. Chimanna(Umrani), A.M.. (2025). EduConvo: AI-Driven Student Feedback System. Journal of Information Systems Engineering and Management. 10.52783/jisem.v10i53s.10861.
10. 223-235.
10. Prado, Victor & Delgado, Carla & Moraes, Laura. (2025). A Self-Assessment for Ethical and Transparent Use of Educational Data. 10.1007/978-3-031-98417-4_19.
11. Patel, Nidhi & Yadav, Bholu. (2025). Code-Collab-Editor. 10.13140/RG.2.2.31273.04961.
12. Mukti, Dwi & Silaban, Juan & D, Raja. (2025). The Influence of Using Moodle-Based Learning Media on Informatics Subjects: Pengaruh Penggunaan Media Pembelajaran Berbasis Moodle pada Mata Pelajaran Informatika. Edu Cendikia: Jurnal Ilmiah Kependidikan. 5. 280-287. 10.47709/educendikia.v5i02.5688.
13. Ezennia, Ikenna & Agbonome, Peter. (2025). Advances in design studio critique techniques: a narrative review on innovative critique models and lessons for improving architecture education. Discover Education. 4. 10.1007/s44217-025-00824-9.
14. Bato, Brittaney & Pomperada, Jake. (2025). Automated grading system with student performance analytics. Technium: Romanian Journal of Applied Sciences and Technology. 30. 58-75. 10.47577/technium.v30i.12871.
15. Karpe, Shivani & Thorve, Anushka & Andhare, Aditi & More, Dhananjay. (2025). A Web Based Centralized System for Academic Performance Tracking.
16. Amanso, Eme Orok & Owan, Valentine. (2025). Ethics In Educational Assessment: Emerging Issues, Controversies, and Best Practices.
17. Mador, Brett. (2025). Competency-Based Education. 10.1007/978-3-031-

86312-7_66.

18. Weegar, Rebecka & Idestam-Almquist, Peter. (2023). Reducing Workload in Short Answer Grading Using Machine Learning. *International Journal of Artificial Intelligence in Education*. 34. 10.1007/s40593-022-00322-1.
19. Lim, Ee & Tsai, Yi-Shan & Maheshi, Bhagya & Nikfalazar, Sanaz. (2025). Enhancing feedback literacy and engagement in project management education: A case study of feedback analytics. *ASCILITE Publications*. 568-573. 10.65106/apubs.2025.2653.
20. Svenningsson, Nina & Faraon, Montathar & Villavicencio, Victor. (2021). Assessment of theses in design education: Conceptualizing a co-creative process for grading by automated criteria evaluation. *International Journal of Innovative Teaching and Learning in Higher Education*. 2. 1-17. 10.4018/IJITLHE.294567.
21. Salisi, Glaizess & Balahadia, Francis. (2025). EduComply: A progress tracking system for compliance in teaching-learning deliverables using decision support system. *International Journal of Science, Technology, Engineering & Mathematics*. 5. 40-78. 10.53378/ijstem.353240.
22. Cardinault, Cynthia & Menéndez Domínguez, Víctor & Aguilar Vera, Raúl. (2016). Interoperability between LMS Moodle and educational specific purpose applications using IMS-LTI services. *Apertura*. 7. 24-34.
23. Barlybayev, Alibek & Razakhova, Bibigul & Sharipbay, Altynbek & Nazyrova, Aizhan & Tursynova, Nazira & Zulkhazhav, Altanbek & Yelibayeva, Gaziza. (2025). Comparative analysis of grading models using fuzzy logic to enhance fairness and consistency in student performance evaluation. *Cogent Education*. 12. 10.1080/2331186X.2025.2481008.
24. Kraus, Kateryna & Kraus, Nataliia & Nikiforov, Petro & Pochenchuk, Galyna & Babukh, Ilona. (2021). Information and Digital Development of Higher Education in the Conditions of Innovatyzation Economy of Ukraine. *WSEAS*

TRANSACTIONS ON ENVIRONMENT AND DEVELOPMENT. 17. 659-671.
10.37394/232015.2021.17.64.

25. Bousboula, Thouaiba & Fadli, Fatima & Khaldi, Mohamed. (2025). Comparative Study between Moodle, Canvas and Google Classroom. International Journal of Research and Innovation in Social Science. IX. 2819-2824. 10.47772/IJRISS.2025.90300219.

26. Huerta, Milagros & Caballero-Hernández, Juan Antonio & Fernández-Ruiz, Manuel. (2022). Comparative Study of Moodle Plugins to Facilitate the Adoption of Computer-Based Assessments. Applied Sciences. 12. 8996. 10.3390/app12188996.

27. Irawan, Rohman & Kuning, Dewi. (2025). Exploring the Implementation of Google Classroom in Online English Language Learning: A Literature Review. Juwara: Jurnal Wawasan dan Aksara. 5. 376-386. 10.58740/juwara.v5i2.658.

28. Hadi, Abil & Jinan, Muthohharun & Maksum, Muh. (2025). The Urgency of Integrating Digital Pedagogy and EdTech in TVET Curriculum to Prepare for Environmentally Friendly Jobs in the 21st Century. ICEETE Conference Series. 3. 204-217. 10.36728/iceete.v3i1.244.

29. Association, Information & Leal, José & Queiros, Ricardo. (2012). A Comparative Study on LMS Interoperability. 10.4018/978-1-4666-0011-9.ch804.

30. Binitie, Amaka & Innocent, Onyemenem & Jigiedous, Okoh. (2025). Enhancing Security and Privacy in EdTech Tools: Safeguarding Student Data in the Digital Learning Era. Journal of Science Innovation and Technology Research. 10.70382/ajsitr.v9i9.041.

31. Lwimba, Evans. (2025). Educational Assessment: How the Assessment Systems can Respond to Diverse Special Needs of the Gifted and Talented in Zambia. 15. 356–370.

32. Inomkhojaeva, Shodiyakhon & Babajanova, Fotima & Abilov, Ural &

- Ongarov, Mansur & Mamaraimova, Zebo & Akobirova, Sarvar & Buriyeva, Gulhayo & Haqqulov, Tohir. (2025). Optimizing Digital Learning with a Comparative Evaluation of Cloud-Based LMS in Higher Education. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*. 16. 10.58346/JOWUA.2025.I3.037.
33. Rojec, Žiga & Puhan, Janez & Fajfar, Iztok. (2025). Practical Programming Exams with Automated Assessment Improve Student Engagement and Learning Outcomes. *Education Sciences*. 15. 1486. 10.3390/educsci15111486.
34. Venkata, Subramanya. (2025). From Code to Cloud: Navigating The Future of SoftwareEngineering and Testing Automation. *International Journal of Basic and Applied Sciences*. 14. 63-70. 10.14419/1zwgxp78.
35. Saragih, Rijoio. (2025). Predicting Student Academic Performance Using Random Forest Regression: A Case Study on LMS Behavioral Data. *International Journal of Information System and Innovative Technology*. 4. 11-14. 10.63322/rm0wcg63.
36. R., Parvathy & Mg, Thushara & Kannimoola, Jinesh. (2025). Automated Code Assessment and Feedback: A Comprehensive Model for Improved Programming Education. *IEEE Access*. PP. 1-1. 10.1109/ACCESS.2025.3554838.
37. Ashfaq, Muhammad. (2025). MACHINE LEARNING FOR PERSONALIZED EDUCATION: A STUDY ON ADAPTIVE LEARNING SYSTEMS. *SSRN Electronic Journal*. 3. 557–567. 10.2139/ssrn/5746242.
38. Yang, Yang. (2025). Test Design for Data Security and Privacy Protection in Training Management Platforms. *Data Ethical and CyberSecurity*. 1. 10.63808/decs.v1i2.79.
39. Harrington, Susan & Klein, Sonja & Fratoni, Alice & Sellberg, Charlott & Lindwall, Oskar. (2025). Meaningful Automated Feedback? Balancing Tensions in Participatory Learning Analytics Design. 10.22318/cscl2025.963119.

40. Mohd Said, Nor. (2025). DEVELOPMENT OF A FUNDAMENTAL PROGRAMMING APP FOR HIGHER EDUCATION LEARNING USING MIT APP INVENTOR 2. 3. 67-75.
41. Spirin, Oleg & Ivanova, Svitlana & Franchuk, Nataliia & Kilchenko, Alla. (2024). ОСНОВНІ СКЛАДНИКИ ЦИФРОВОЇ КОМПЕТЕНТНОСТІ НАУКОВИХ І НАУКОВО-ПЕДАГОГІЧНИХ ПРАЦІВНИКІВ ЗАКЛАДІВ ВИЩОЇ ОСВІТИ УКРАЇНИ. UNESCO Chair Journal Lifelong Professional Education in the XXI Century. 2. 91-103. 10.35387/ucj.2(10).2024.0007.
42. Kaleci, Devkan. (2025). Integration and application of artificial intelligence tools in the Moodle platform: A theoretical exploration. Journal of Educational Technology and Online Learning. 8. 100-111. 10.31681/jetol.1595079.
43. Garai, Pooja. (2025). Envision EdTech: Revolutionizing Intelligent Education through AI and Innovation for a Smarter Tomorrow. International Journal of Innovative Science and Research Technology. 3095-3110. 10.38124/ijisrt/25apr1980.
44. Guntara, Ranga & Azkarin, Varinia. (2023). Pembangunan REST API Human Resource Information System Domain Pengelolaan User Dengan Menggunakan Framework Express JS dan Node.js. ULIL ALBAB : Jurnal Ilmiah Multidisiplin. 2. 3932-3944. 10.56799/jim.v2i8.2070.
45. Mozzherin, Dmitry & Ower, Geoffrey. (2024). SQLite: A “Frictionless” Solution for Exchange of Biodiversity Data?. Biodiversity Information Science and Standards. 8. 10.3897/biss.8.138931.
46. Shcheblanin, Yurii & Sydorenko, Bohdan & Mykhalchuk, Inna. (2024). SECURITY OF REST API: THREATS AND PROTECTION METHODS. Information systems and technologies security. 2. 60-65. 10.17721/ISTS.2024.8.60-65.

47. Thakur, Lalman & Singh, Joybir & Chahar, Nikhil & Singh, Dr. (2025). An Integrated Learning System to Transform Education: A Full Stack Web Development Approach. 10.2139/ssrn.5239700.
48. Forment, Marc & Pereira, Juanan & García-Peñalvo, Francisco & Casañ, Maria & Cabré, Jose. (2025). LAMB: An open-source software framework to create artificial intelligence assistants deployed and integrated into learning management systems. *Computer Standards & Interfaces*. 92. 103940. 10.1016/j.csi.2024.103940.
49. Micheal, Dave. (2025). Enhancing User Interfaces with React Query and Material-UI Integration.
50. Riady, Agil & Paniran, Paniran & Suksmadana, I. (2024). Perancangan Backend Api Berbasis Rest-API pada Aplikasi Rekomendasi Resep Makanan. *Mars : Jurnal Teknik Mesin, Industri, Elektro Dan Ilmu Komputer*. 2. 94-106. 10.61132/mars.v2i3.137.
51. Borovskoy, I & Shelmina, E & Zarikovskaya, N. (2017). Text Data Storage Method Modification for SQLite Relational Database. *Journal of Physics: Conference Series*. 803. 012025. 10.1088/1742-6596/803/1/012025.
52. Tanveer, Fatima & Iradat, Faisal & Iqbal, Waseem & Ahmad, Awais. (2025). Towards Secure APIs: A Survey on RESTful API Vulnerability Detection. *Computers, Materials & Continua*. 84. 4223-4257. 10.32604/cmc.2025.067536.
53. Wanner, Lucas. (2024). Integrating Continuous Assessment into Undergraduate Computer Architecture using Automated Grading. *International Journal of Computer Architecture Education*. 13. 24-32. 10.5753/ijcae.2024.5341.
54. Talaseela, Rahul. (2025). Modular Extensibility in Enterprise Systems: Enabling Business Agility through Layered Architecture. *International Journal of Computational and Experimental Science and Engineering*. 11. 10.22399/ijcesen.3861.

55. Soyemi, Jumoke & Ige, Ogunyinka & Soyemi O.B, Jide Soyemi & Ademola, Ajibodu & Jayeoba, Adaramola & Olumide, Afolayan & Amode, Habeeb & Akinde, Mukail. (2025). Design and development of home-grown biometric fingerprint device and software for attendance and access control. *International Journal of Electrical and Computer Engineering (IJECE)*. 15. 5616. 10.11591/ijece.v15i6.pp5616-5632.
56. Smolentseva, Tatyana. (2025). Designing a modular automated decision support information system in a digital educational environment. *Computational nanotechnology*. 12. 141-151. 10.33693/2313-223X-2025-12-3-141-151.
57. Kalra, Shairy & Arora, Gourav. (2025). Synergy between Relational and NOSQL Databases for Big Data Management and Shaping the Future of Database Education. *International Journal of Computer Applications*. 187. 26-31. 10.5120/ijca2025924923.
58. Munagandla, Vamshi Bharath & Dandyala, Varshika & Vadde, Bharath Chandra. (2023). Leveraging Cloud Data Integration for Enhanced Learning Analytics in Higher Education. 3.
59. Quiban, Josie. (2025). Modeling Faculty Acceptance of LMS: A PLS-SEM Validation of the Technology Acceptance Model in Philippine Higher Education. *Business, Education, Social Sciences, and Technology*. 1. 103-108. 10.69478/BEST2025v1n1a016.
60. Hasan, Mohammad. (2024). *Web Programming Lab Manual*. 10.13140/RG.2.2.28616.12802.
61. Sharma, Vaishnavi & Thakur, Rakesh & Sharma, Shashwat & Panjanani, Kritika. (2025). Artificial Intelligence-Powered Assessment Framework for Skill-Oriented Engineering Lab Education. 10.48550/arXiv.2509.25258.
62. (2025). Enhancing Software Quality and Efficiency: The Role of Generative AI in Automated Code Generation and Testing.

63. Koo, Jiyoung & Jo, Il-Hyun. (2025). A Theoretical Framework for User Mental Models in Learning Analytics Systems. Korea Institute of Design Research Society. 10. 9-21. 10.46248/kidrs.2025.3.9.
64. Mironov, Valeriy & Mironov, Konstantin. (2025). AM-integrity in designing relational database models. *Ontology of Designing*. 15. 552-565. 10.18287/2223-9537-2025-15-4-552-565.
65. Amin, Waqas & Kanakasabai, Saravanan & Grout, Randall & Butler, Joe & Michael, Scott & Schleyer, Titus. (2025). 348 Real-time data synchronization: Assessing the implementation of REDCap CDIS (Clinical Data Interoperability Service) for EHR systems. *Journal of Clinical and Translational Science*. 9. 107-107. 10.1017/cts.2024.975.
66. Jayakody, Dineth & Isuranda, Koshila & A.V.A, Malkith. (2025). MCQ Automated Grading Using Image Processing Techniques. 10.13140/RG.2.2.27089.03683.
67. Azuddin, Muna & Yusup, Muhamad & Setiyowati, Harlis & Wibowo, Shesilia & Suwita, Jaka & Basuki, Sucipto & Astuti, Eka. (2025). Design and Implementation of a Relational Database for an Academic Information System. *International Transactions on Artificial Intelligence (ITALIC)*. 3. 171-180. 10.33050/italic.v3i2.801.
68. Muayyadin, Muhammad & Irfan, Pahrul & Rassy, Regania. (2025). Frontend Implementation of UI/UX Design of Sumbawa District Agricultural Service Website using React JS Framework: IMPLEMENTASI FRONTEND DARI DESIGN UI/UX WEBSITE DINAS PERTANIAN KABUPATEN SUMBAWA MENGGUNAKAN FRAMEWORK REACT JS. *Jurnal Begawe Teknologi Informasi (JBegaTI)*. 6. 10.29303/jbegati.v6i1.1326.
69. Flick, Taylor. (2025). Best Practices for Dashboard UI/UX to Enhance User Experience.

70. Anwar, Fahmi & Fadlil, Abdul & Riadi, Imam. (2020). Validation Analysis of Scalable Vector Graphics (SVG) File Upload using Magic Number and Document Object Model (DOM). *International Journal of Advanced Computer Science and Applications*. 11. 255-262. 10.14569/IJACSA.2020.0111133.
71. Clinton, Ronald & Steve, Dorcas & Eric, Tesco. (2025). Revolutionizing Student Assessment: How NLP Can Enhance Grading and Feedback Author.
72. Yusupova, Regina & Zaripova, Rimma & Agafonova, Kristina. (2025). CREATING A DASHBOARD FOR ANALYTICS AND VISUALIZATION OF EDUCATIONAL PROCESS INDICATORS. *EKONOMIKA I UPRAVLENIE: PROBLEMY, RESHENIYA*. 9/5. 123-131. 10.36871/ek.up.p.r.2025.09.05.016.
73. Haksa, Febrina & Endri. (2025). Interactive Real-Time Weight Management Platform Using Machine Learning Methods. *Indonesian Journal of Artificial Intelligence and Data Mining*. 8. 397. 10.24014/ijaidm.v8i2.36874.
74. Lee, Jay & Ahn, Joongwon & Yi, Kwangkeun. (2025). React-tRace: A Semantics for Understanding React Hooks: An Operational Semantics and a Visualizer for Clarifying React Hooks. *Proceedings of the ACM on Programming Languages*. 9. 471-498. 10.1145/3763067.
75. (2025). RESTful API Design for Geospatial Logistics Platforms Using Type Script and Laravel. *Journal of Information, Technology and Policy*. 1-13. 10.62836/jitp.2025.515.
76. Parlakkılıç, Alaattin. (2019). Responsive Mobile Learning (M-Learning) Application Design And Architecture In Fog Computing. *International Journal of Modern Education Studies*. 3. 82. 10.51383/ijonmes.2019.40.
77. Fitriyani, Yuli & Syahbana. (2025). Interactive Dashboard for Visualizing Teacher Distribution in Tangerang City Using BigQuery and Looker Studio. *Journal of Multidisciplinary Research and Technology*. 1. 39-50. 10.63891/j-

mart.v1i2.62.

ДОДАТОК А

СТРУКТУРА БАЗИ ДАНИХ

А.1. Загальна схема

База даних реалізована на основі SQLite і містить три основні таблиці:

- users — інформація про користувачів
- labs — перелік лабораторних робіт
- submissions — оцінки студентів

А.2. SQL-структура

-- Таблиця користувачів

```
CREATE TABLE users (  
  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
  
    login TEXT NOT NULL UNIQUE,  
  
    password TEXT NOT NULL,  
  
    full_name TEXT NOT NULL,  
  
    role TEXT CHECK(role IN ('student', 'teacher')) NOT NULL  
  
);
```

-- Таблиця лабораторних робіт

```
CREATE TABLE labs (  
  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
  
    title TEXT NOT NULL,
```

```
description TEXT

);

-- Таблиця оцінок

CREATE TABLE submissions (

    user_id INTEGER,

    lab_id INTEGER,

    score INTEGER CHECK(score BETWEEN 0 AND 5),

    PRIMARY KEY (user_id, lab_id),

    FOREIGN KEY (user_id) REFERENCES users(id),

    FOREIGN KEY (lab_id) REFERENCES labs(id)

);
```

A.3. Початкове наповнення

```
-- Додавання викладача

INSERT INTO users (login, password, full_name, role)

VALUES ('teacher1', 'pass123', 'Іваненко Іван Іванович', 'teacher');

-- Додавання студентів

INSERT INTO users (login, password, full_name, role)

VALUES
```

```
('stud1', 'pass1', 'Петренко Петро', 'student'),
```

```
('stud2', 'pass2', 'Коваль Олена', 'student');
```

-- Додавання лабораторних

```
INSERT INTO labs (title, description)
```

```
VALUES
```

```
('Лабораторна №1', 'Оператори та вирази'),
```

```
('Лабораторна №2', 'Умовні конструкції'),
```

```
('Лабораторна №3', 'Цикли'),
```

```
('Лабораторна №4', 'Функції'),
```

```
('Лабораторна №5', 'Робота з файлами');
```

A.4. Особливості реалізації

- Поле `score` дозволяє зберігати оцінку в діапазоні від 0 до 5.
- Первинний ключ у таблиці `submissions` — комбінація `user_id` та `lab_id`, що гарантує унікальність оцінки для кожної лабораторної роботи кожного студента.
- Всі зовнішні ключі забезпечують цілісність даних.

ДОДАТОК Б

ФРОНТЕНД ПРОГРАМНОЇ СИСТЕМИ

Б.1. Компонент LoginPage

```
import React, { useState } from 'react';

import {

  Container,

  TextField,

  Button,

  Typography,

  Alert,

} from '@mui/material';

import { api } from '../api';

export default function LoginPage({ onLogin }) {

  const [login, setLogin] = useState("");

  const [password, setPassword] = useState("");

  const [error, setError] = useState("");

  const handleLogin = async () => {

    try {
```

```

const res = await api.post('/auth/login', { login, password });

if (res.data.success) {

    onLogin(res.data.user);

} else {

    setError('Невірний логін або пароль');

}

} catch {

setError('Помилка з'єднання з сервером');

}

};

return (

<Container maxWidth="xs" sx={{ mt: 8 }}>

<Typography variant="h5" gutterBottom>

    Вхід до системи

</Typography>

{error && <Alert severity="error">{error}</Alert>}

<TextField

    label="Логін"

    fullWidth

```

```

        margin="normal"

        value={login}

        onChange={(e) => setLogin(e.target.value)}

    />

    <TextField

        label="Пароль"

        type="password"

        fullWidth

        margin="normal"

        value={password}

        onChange={(e) => setPassword(e.target.value)}

    />

    <Button variant="contained" fullWidth sx={{ mt: 2 }} onClick={handleLogin}>

        Увійти

    </Button>

</Container>

);

}

```

Б.2. Компонент StudentDashboard

```
import React, { useEffect, useState } from 'react';
```

```
import {  
  Container,  
  Typography,  
  Table,  
  TableBody,  
  TableCell,  
  TableHead,  
  TableRow,  
  Button,  
} from '@mui/material';  
  
import { api } from '../api';  
  
export default function StudentDashboard({ onLogout }) {  
  const [labs, setLabs] = useState([]);  
  
  useEffect(() => {  
    api.get('/labs/student-status').then((res) => setLabs(res.data));  
  }, []);  
  
  return (  

```

```
<Container sx={{ position: 'relative', mt: 2 }}>
```

```
<Button
```

```
  variant="outlined"
```

```
  color="error"
```

```
  onClick={onLogout}
```

```
  sx={{ position: 'absolute', top: 0, right: 0 }}

```

```
>
```

```
  Вихід
```

```
</Button>
```

```
<Typography variant="h5" gutterBottom>
```

```
  Кабінет студента
```

```
</Typography>
```

```
<Table size="small">
```

```
  <TableHead>
```

```
    <TableRow>
```

```
      <TableCell>Лабораторна</TableCell>
```

```
      <TableCell>Оцінка</TableCell>
```

```
    </TableRow>
```

```

</TableHead>

<TableBody>

{labs.map((lab) => (

<TableRow key={lab.lab_id}>

<TableCell>{lab.title}</TableCell>

<TableCell>{lab.score ?? '—'}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</Container>

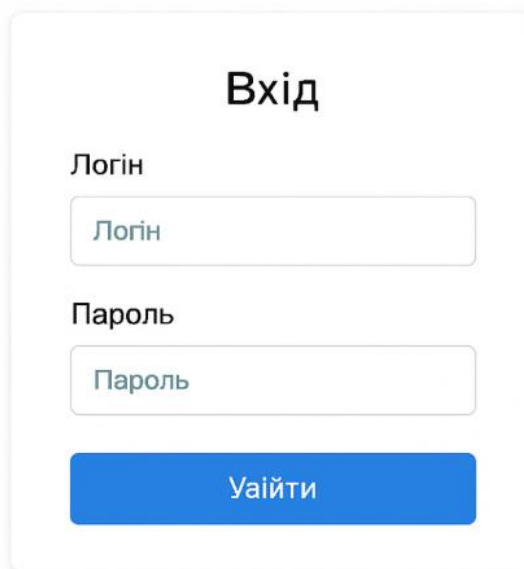
);

}

```

Ось згенерований скриншот інтерфейсу викладача для оцінювання лабораторних робіт. Він ілюструє таблицю з іменами студентів, назвами лабораторних, полями для виставлення оцінок (0–5) та підсумковим балом, який підсвічується зеленим, якщо всі роботи оцінено:

Система підтримки лабораторних робіт



Вхід

Логін

Логін

Пароль

Пароль

Увійти

ДОДАТОК В

СКРІНШОТИ ІНТЕРФЕЙСУ

Б.1. Сторінка входу

- **Заголовок:** «Система підтримки лабораторних робіт»
- **Поле логіну:** введення імені користувача
- **Поле паролю:** захищене введення
- **Кнопка «Увійти»:** авторизація користувача

Б.2. Кабінет викладача

- **Стовпець «Студент»:** ПІБ студента
- **Стовпці лабораторних:** назви робіт (Лаб. №1, №2...)
- **Оцінки:** випадуючі списки з балами 0–5
- **Загальний бал:** сума / середній бал, підсвічується зеленим, якщо всі оцінки виставлено
- **Кнопка «Вихід»:** завершення сесії

ДОДАТОК Г

БЕКЕНД-КОД ТА ОПИС API

Г.1. Загальна структура

Бекенд реалізовано на Node.js з використанням Express. Основні маршрути забезпечують авторизацію, завантаження лабораторних робіт, оновлення оцінок та отримання статусу студентів.

Г.2. Основні маршрути

```
const express = require('express');

const sqlite3 = require('sqlite3').verbose();

const multer = require('multer');

const app = express();

const upload = multer({ dest: 'uploads/' });

app.use(express.json());
```

```
const db = new sqlite3.Database('database.db');
```

□ **POST /auth/login**

Авторизація користувача за логіном і паролем.

```
app.post('/auth/login', (req, res) => {

  const { login, password } = req.body;

  db.get(

    'SELECT * FROM users WHERE login = ? AND password = ?',

    [login, password],

    (err, user) => {

      if (user) {

        res.json({ success: true, user });

      } else {

        res.json({ success: false });

      }

    }

  );

});
```

□ **POST /labs/upload**

Завантаження лабораторної роботи студентом.

```
app.post('/labs/upload', upload.single('file'), (req, res) => {
```

```
const { user_id, lab_id } = req.body;

// Файл збережено у req.file.path

res.json({ success: true });

});
```

□ **POST /labs/update-score**

Оновлення оцінки викладачем.

```
app.post('/labs/update-score', (req, res) => {

  const { user_id, lab_id, score } = req.body;

  db.run(

    'INSERT OR REPLACE INTO submissions (user_id, lab_id, score) VALUES (?, ?, ?)',

    [user_id, lab_id, score],

    function (err) {

      res.json({ success: !err });

    }

  );

});
```

□ **GET /labs/students-status**

Отримання оцінок усіх студентів з підсумками.

```
app.get('/labs/students-status', (req, res) => {
```

```

db.all('SELECT * FROM users WHERE role = "student"', [], (err, students) => {

const result = [];

students.forEach((student) => {

db.all(

  `SELECT labs.id AS lab_id, labs.title, submissions.score

  FROM labs LEFT JOIN submissions

  ON labs.id = submissions.lab_id AND submissions.user_id = ?`,

  [student.id],

  (err, labs) => {

const scores = labs.map((l) => l.score).filter((s) => s !== null);

const sum = scores.reduce((a, b) => a + b, 0);

const avg = scores.length ? (sum / scores.length).toFixed(2) : 0;

result.push({

  id: student.id,

  full_name: student.full_name,

  labs,

  sum,

  average: avg,

});

if (result.length === students.length) res.json(result);

```

```
    }  
  
);  
  
});  
  
});  
  
});
```

ДОДАТОК Д

ІНСТРУКЦІЯ КОРИСТУВАЧА

Д.1. Інструкція для студента

☐ Авторизація

1. Відкрийте головну сторінку системи.
2. У полі «Логін» введіть свій логін.
3. У полі «Пароль» введіть пароль.
4. Натисніть кнопку «**Увійти**».

☐ Перегляд лабораторних робіт

1. Після входу відкриється **Кабінет студента**.
2. У таблиці відображаються всі лабораторні роботи, призначені для виконання.
3. У стовпці «Оцінка» вказано вашу оцінку (0–5) або «—», якщо оцінку ще не виставлено.

☐ Завантаження файлу (якщо реалізовано)

1. Натисніть кнопку «Завантажити» біля відповідної лабораторної.
2. Виберіть файл зі свого пристрою.
3. Натисніть «Підтвердити» — файл буде надіслано викладачу.

☐ **Вихід із системи**

1. Натисніть кнопку «**Вихід**» у правому верхньому куті.

Д.2. Інструкція для викладача

☐ **Авторизація**

1. Введіть логін і пароль викладача.
2. Натисніть кнопку «**Увійти**».

☐ **Перегляд студентів**

1. Після входу відкриється **Кабінет викладача**.
2. У таблиці відображаються всі студенти та їх лабораторні роботи.

☐ **Виставлення оцінок**

1. У кожній клітинці лабораторної роботи виберіть оцінку зі списку (0–5).
2. Після вибору оцінка автоматично зберігається.
3. Якщо всі оцінки виставлено — підсумковий бал підсвічується зеленим.

☐ **Підсумкові показники**

- У останньому стовпці таблиці відображається:
 - **Сума балів** — загальна кількість отриманих балів.
 - **Середній бал** — середнє арифметичне всіх оцінок.

□ **Вихід із системи**

1. Натисніть кнопку «**Вихід**» у правому верхньому куті.