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

ПРОГРАМНЕ СЕРЕДОВИЩЕ НАВЧАЛЬНОГО ПРИЗНАЧЕННЯ «БУЛЕВІ ФУНКЦІЇ ТА СХЕМИ З ФУНКЦІОНАЛЬНИХ ЕЛЕМЕНТІВ»

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконала: студентка 4 курсу 441 групи
Спеціальності: 121 Інженерія
програмного забезпечення
Освітньо-професійної програми:
Інженерія програмного забезпечення
Руда Ю.Ю.
Керівник: професор ККНПІ
Львов М.С.
Рецензент: доцент КАГМА
Котова О.В.

ХЕРСОН – 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application Programming Interface
CSRF	–	Cross-Site Request Forgery
CSS	–	Cascading Style Sheets
HTML	–	HyperText Markup Language
ORM	–	Object-Relational Mapping
SEO	–	Search Engine Optimization
SQL	–	Structured Query Language
SSG	–	Static Site Generators
SSR	–	Server-Side Rendering
БД	–	База даних
СУБД	–	Система управління базами даних

ВСТУП

Актуальність теми. Розробка програмного середовища навчального призначення для вивчення булевих функцій та схем функціональних елементів є дуже актуальною в наш час, коли відбувається швидкий технологічний розвиток і дедалі більше професій вимагають високого рівня комп'ютерної грамотності та інших цифрових навичок.

Особливо важливою є розробка програм навчання для шкільної та університетської освіти, оскільки вона повинна забезпечувати підготовку молодих людей до роботи в умовах сучасного світу. Також, програми навчання можуть бути корисними для працівників, які бажають здобути нові знання та навички в певній області.

Розробка програм навчання може допомогти покращити якість освіти та забезпечити більш ефективний процес навчання. Вони можуть бути використані для автоматизації рутинних завдань, підвищення зацікавленості та мотивації учнів, сприяння індивідуальному навчанню та багато іншого.

Отже, розробка програм навчання є дуже важливою та актуальною, оскільки може допомогти підготувати наступне покоління до викликів сучасного світу та забезпечити їм необхідні навички та знання.

Мета і завдання дослідження. Метою дослідження є розробка програмного середовища навчального призначення для вивчення булевих функцій та схем функціональних елементів. Завдання дослідження:

- провести аналіз предметної області;
- провести аналіз існуючих програм;
- спроектувати програмне середовище;
- розробити програму;
- провести тестування програмного середовища.

Об’єкт дослідження: дисципліна «Дискретна математика», вивчення булевих функцій та схеми з функціональних елементів.

Предмет дослідження: програмне середовище навчального призначення для вивчення булевих функцій та схем з функціональних елементів.

Методи дослідження. Отримані результати дослідження базуються на методах спостереження, порівняння, системного аналізу та дедукції.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД АНАЛОГІВ ПРОГРАМНИХ СЕРЕДОВИЩ НАВЧАЛЬНОГО ПРИЗНАЧЕННЯ

1.1 Визначення поняття «булева функція», схеми функціональних елементів

Булеві функції – це функції, які приймають один або більше булевих (логічних) аргументів, тобто змінних, які можуть мати значення «істина» (*true*) або «хиба» (*false*), і повертають булеве значення (*true* або *false*) на основі цих аргументів. Вони названі на честь алгебричного вченого Джорджа Буля, який розробив алгебру логіки

Булеві функції є важливими для логіки і комп'ютерних наук, оскільки вони можуть бути використані для визначення логічних відносин між різними станами і подіями. Наприклад, можна використовувати булеві функції для визначення, чи потрібно виконувати певний код, якщо певний умовний оператор повертає *true*, або для визначення, чи знаходиться даний елемент у масиві [1].

Булеві функції та схеми функціональних елементів вивчаються в рамках дисципліни «Дискретна математика» [2]. Дискретна математика – це розділ математики, який вивчає математичні структури, що складаються з окремих елементів, а не з безперервних величин. Вона зосереджена на алгоритмах, графах, теорії чисел, теорії комбінаторики та теорії груп.

Математичний апарат дискретної математики дуже важливий для розв'язання багатьох практичних проблем в сучасній науці та технологіях, таких як теорія обчислювальних процесів, теорія інформації, теорія кодування, криптографія, мережі та ін.

Основні поняття, які використовуються в дискретній математиці, включають:

– множини – набір елементів без порядку або повторення.

- реляції – взаємовідношення між елементами множин.
- графи – абстрактні структури, що складаються з вершин та ребер, які з'єднують вершини.
- функції – правила, які встановлюють відповідність між елементами множин.
- логіка – розділ математики, що вивчає принципи розв'язання задач, заснованих на правилах індукції та дедукції.

Ці поняття є основними будівельними блоками дискретної математики, і вони використовуються для вивчення різних тем, таких як теорія графів, теорія кодування та теорія комбінаторики.

Також в дискретній математиці використовуються різні методи та інструменти для аналізу та розв'язання проблем, такі як алгоритми, формальна логіка, теорія ймовірностей, теорія чисел та інші.

Загалом, математичний апарат дискретної математики є незамінним інструментом для розв'язання задач, пов'язаних з обробкою даних та інформацією, проектуванням та оптимізацією алгоритмів, а також розробкою та аналізом комп'ютерних систем.

У контексті вивчення булевих функцій, дискретна математика допомагає в розумінні логічної структури булевих функцій та розвитку методів для їх аналізу та оптимізації. Наприклад, теорія графів використовується для представлення та аналізу великих множин булевих функцій, а теорія комбінаторики дозволяє знайти кількість можливих комбінацій для заданих вхідних значень.

Зокрема, в дискретній математиці існує також теорія булевих функцій, яка вивчає булеві функції як математичні об'єкти та їх властивості. Вона зосереджена на дослідженні логіки, алгебри, топології та інших галузях математики в контексті булевих функцій.

Теорія булевих функцій також використовується для дизайну та аналізу цифрових схем, що мають важливе значення для розробки електронних пристроїв, таких як комп'ютери, мобільні пристрої та інші.

Отже, математичний апарат дискретної математики дозволяє не тільки розуміти теоретичні основи булевих функцій, але й застосовувати їх для вирішення різноманітних завдань у сучасному світі.

Вивчення булевих функцій включає розуміння основних концепцій булевої алгебри, таких як булеві змінні, операції AND, OR, NOT, еквівалентність та імплікація, а також їх застосування в різних областях, таких як електроніка, комп'ютерні науки, математика та логіка.

Основні кроки для вивчення булевих функцій:

1. Ознайомитися з базовими поняттями булевої алгебри, такими як булеві змінні, таблиці істинності та основні булеві операції.
2. Розглянути різні типи булевих функцій, включаючи кон'юнкцію, диз'юнкцію, заперечення, еквівалентність та імплікацію.
3. Вивчити основні правила булевої алгебри, такі як дистрибутивність, асоціативність та комутативність.
4. Вивчити методи спрощення булевих функцій, такі як метод Карно, метод Булевої алгебри та інші.
5. Ознайомитися з застосуванням булевих функцій у логіці, математиці, електроніці та комп'ютерних науках.
6. Виконати практичні завдання зі складання булевих виразів та спрощення булевих функцій.
7. Дослідити застосування булевих функцій у різних областях, таких як автоматичне керування, шифрування, дизайн інтерфейсу та багато інших.
8. Вивчити поняття квантових булевих функцій та їх застосування у квантових обчисленнях.

Загалом, вивчення булевих функцій є важливим елементом для будь-якої людини, яка працює у галузі науки, технологій та інформаційних систем. Булеві функції є основою для багатьох систем, зокрема для електронних пристроїв, програмного забезпечення та мереж.

Вивчення булевих функцій допоможе розвинути критичне мислення, логічне мислення та аналітичні здібності. Воно також може бути корисним для підготовки до спеціалізованих курсів, таких як теорія автоматів, теорія обчислювальних процесів та дизайн логічних схем.

Основними джерелами інформації для вивчення булевих функцій є підручники з логіки та теорії обчислювальних процесів, онлайн-курси та відео-уроки, а також спеціалізовані форуми та спільноти. Для практичного використання булевих функцій можна використовувати спеціалізовані програми та середовища розробки, такі як MATLAB, Octave, Python та інші.

Основним поняттям алгебри логіки є висловлення. Образно кажучи, висловлення – це деяке твердження, про яке можна сказати, що воно є істинним або хибним. Будь-яке висловлення можна позначити символом x і вважати, що $x = 1$ при істинності, а $x = 0$ при хибності висловлення. Логічними (булевими) змінними в булевій алгебрі називають величини, які незалежно від їхньої конкретної суті можуть набувати лише двох значень. Ці значення позначаються нулем (0) й одиницею (1). Якщо змінна x має одиничне значення, то записуємо $x = 1$, а якщо нульове, то $x = 0$ [3].

Існує кілька способів задання булевих функцій:

- Таблиця істинності (таблиця 1.1): Цей метод полягає в складанні таблиці, яка відображає всі можливі комбінації значень вхідних аргументів та відповідні результати функції. Це найбільш універсальний спосіб задання булевих функцій і

можна використовувати для будь-якої функції з будь-якою кількістю аргументів [4].

- Булева формула: Це алгебраїчний підхід до задання булевих функцій, в якому використовуються логічні оператори (AND, OR, NOT) та змінні, щоб скласти алгебраїчний вираз, який описує функцію [5].
- Графічне представлення: Графічне представлення булевих функцій використовує графіки, діаграми або схеми, щоб показати логічні відносини між змінними [6].
- Мінімізація Карно: Цей метод заснований на використанні діаграм Карно, які зображують всі можливі комбінації значень вхідних аргументів та відповідні результати функції у вигляді таблиці. За допомогою мінімізації Карно можна спростити складні булеві функції та скоротити їхню реалізацію [7].

Таблиця 1.1

Аргументи		$\bar{A}$	$\bar{B}$	$A \wedge B$	$A \vee B$	$A \leftrightarrow B$
A	B					
0	0	1	1	0	0	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	1	1	1

- $\bar{A}$ – заперечення A (логічне NOT);
- $\bar{B}$ – заперечення B (логічне NOT);
- $A \wedge B$ – кон’юнкція (логічне AND);
- $A \vee B$ – диз’юнкція (логічне OR);
- $A \leftrightarrow B$ – еквівалентність (логічне XNOR).

Нехай деякий пристрій дискретної дії має певну кількість пронумерованих входів та деяку кількість виходів. На кожен із входів може подаватись сигнал, який приймає значення 0 або 1 і таким чином, що набір сигналів на вході визначає набір сигналів на виході, то такий пристрій буде називатись функціональним елементом. А сукупність цих елементів і їх композиція є схемою з функціональних елементів [8].

Схема з функціональних елементів – математичний об'єкт. Очевидно, що схема з функціональних елементів реалізує певну булеву функцію. Оскільки допустимими є з'єднання елементів, які відповідають суперпозиціям відповідних елементарних функцій, а система є функціонально повною, то будь-яку функцію можна реалізувати схемою з функціональних елементів, зображених на рисунку 1.1.

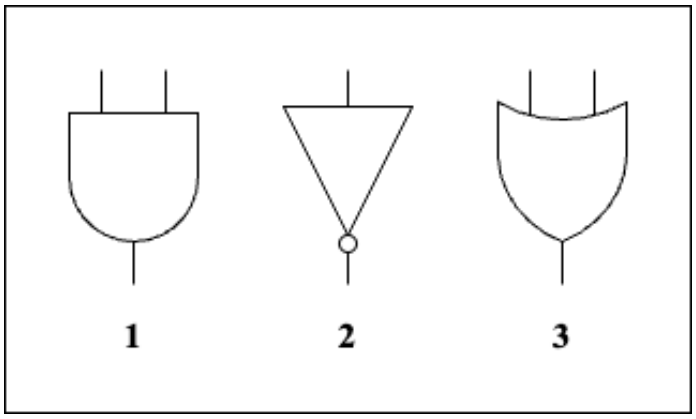

Рисунок 1.1 – Функціональні елементи
1 – кон’юнкція (AND), 2 – заперечення (NOT), 3 – диз’юнкція (OR)

Задача синтезу схем функціональних елементів полягає в тому, щоб знайти таку комбінацію базових логічних елементів (наприклад, AND, OR, NOT), яка реалізує задану булеву функцію.

Один з найпоширеніших методів синтезу схем полягає в застосуванні методу Карно. Цей метод використовує таблиці істинності для знаходження мінімальної комбінації логічних елементів, яка реалізує

задану функцію. Застосування методу Карно дозволяє значно зменшити кількість логічних елементів у схемі та збільшити її швидкодію.

Задача аналізу схем функціональних елементів полягає в тому, щоб досліджувати властивості та поведінку схем, побудованих з базових логічних елементів (наприклад, AND, OR, NOT). Один з методів аналізу схем – це метод аналізу таблиці істинності. Цей метод використовує таблиці істинності, щоб знайти відповідність між входами та виходами схеми. В результаті аналізу таблиці істинності можна визначити, чи правильно працює схема при різних комбінаціях входів, а також можна знайти значення вхідних змінних, що призводять до виведення певних значень на виході.

У загальному, задача аналізу схем функціональних елементів допомагає розуміти, як працюють електронні пристрої та як їх можна покращувати, а також забезпечує необхідні знання для проектування ефективних електронних пристроїв у майбутньому.

Задача оптимізації схем функціональних елементів полягає в зменшенні кількості елементів, які необхідні для заданої функції, зменшенні затрат на виготовлення, та забезпеченні максимальної ефективності і надійності схеми.

Один із способів подання логічних функцій – карти Карно. В карті Карно булеві змінні переносяться (зазвичай з таблиці істинності) і впорядковуються згідно з принципами кода Грея, в якому тільки одна змінна змінюється при переході між сусідніми квадратами. Коли таблиця згенерована, і у відповідні комірки записані вихідні значення, дані організовуються в найбільші можливі групи. Далі, працюючи з цими групами, отримують мінімізовану диз’юнктивну нормальну форму. Нижче представлено приклад мінімізації функції 1.1 методом Карно.

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} \quad (1.1)$$
$$+ A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}\bar{D} + ABCD$$

Наносимо цю функцію на карту Карно (рисунок 1.2) і проводимо можливі контури.

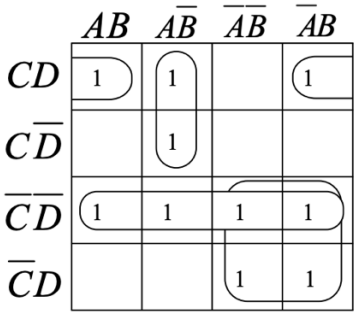

Рисунок 1.2 – Мінімізація булевої функції 1.1

Використовуючи закон склеювання для кожного контурі записують мінімальний булевий вираз і таким чином отримують мінімальну булеву функцію. Після склеювання із чотирьох контурів отримаємо значно простішу мінімальну булеву функцію 1.2.

$$F = \bar{A}\bar{C} + \bar{C}\bar{D} + BCD + A\bar{B}C \quad (1.2)$$

Карта Карно зазвичай стає важкою для розпізнання при збільшені кількості змінних. Загальне правило таке, карта Карно добре працює до чотирьох-п'яти змінних, і не має використовуватись з більше ніж шістьома змінними. Для виразів з більшою кількістю змінних може бути використаний Метод Квайна [7].

1.2 Аналіз предметної області та аналіз існуючих програмних середовищ навчального призначення

Традиційна методика вивчення булевих функцій включає наступні етапи:

1. Вивчення логіки – логіка є основою булевих функцій. У цьому етапі ви дізнаєтеся про принципи логіки, які використовуються у булевих функціях, такі як логічні операції, таблиці істинності та інші.
2. Вивчення булевих функцій – на цьому етапі ви ознайомитесь з основними булевими функціями, такими як AND, OR, NOT,

XOR тощо. Ви дізнаєтеся, як вони працюють, та як вони можуть бути складені разом, щоб створити складніші функції.

3. Вивчення алгебри булевих функцій – на цьому етапі ви дізнаєтеся про алгебру булевих функцій, яка використовується для розв'язання складних задач. Ви навчитеся використовувати закони алгебри булевих функцій, щоб зменшити кількість логічних операцій, які потрібно виконати для досягнення бажаного результату.
4. Розв'язання практичних задач – на цьому етапі ви застосуєте свої знання для розв'язання практичних задач, таких як дизайн логічних схем, розробка програмного забезпечення та різних електронних пристроїв.
5. Продовження вивчення – на цьому етапі ви будете вивчати більш складні булеві функції, а також спеціалізовані теми, такі як теорія автоматів, теорія обчислювальних процесів та інші.

Традиційна методика вивчення булевих функцій передбачає вивчення теорії та її практичного використання. Вона може бути корисною для тих, хто вивчає булеві функції вперше і має бажання поглибити свої знання у цій області. Також ця методика допоможе вам зрозуміти логіку роботи багатьох електронних пристроїв, таких як комп'ютери, мікроконтролери, логічні вентилі та інші.

Однак, традиційна методика вивчення булевих функцій може бути недостатньою для тих, хто хоче займатися розвитком програмного забезпечення, дизайном апаратних засобів або дослідженням в галузі теорії обчислювальних процесів. У цьому випадку потрібне більш глибоке вивченням теорії булевих функцій, алгоритмів та структур даних, що використовують булеві функції.

Крім традиційної методики, існують і альтернативні методики вивчення булевих функцій, такі як онлайн-курси, спеціалізовані книги та

відеоуроки. Вони можуть бути корисними для тих, хто більше зацікавлений у практичному застосуванні булевих функцій, або хто має обмежений час для вивчення теорії.

Серед різних програмних продуктів, що забезпечують навчальний процес особливе місце займають саме програми-тренажери, які дозволяють сформувати у студентів практичні навички на основі вивченого теоретичного матеріалу [9].

Дослідженнями в сфері розробки додатків та програм для вивчення дискретної математики та булевих функцій займалися велика кількість вчених. Ерендіра М. Хіменес-Ернандес, Ганна Октаба, Фріда Діас-Барріга, Маріо П'яттіні в своїй роботі [10] представляють програмне забезпечення під назвою MiniBool, яке було розроблено з метою підтримки вивчення булевої алгебри в умовах змішаного навчання. Ця освітня пропозиція дає учням можливість посилити навчання в будь-який час і в будь-якому місці. Вона додатково підвищує мотивацію учнів шляхом включення гейміфікації через використання рейтингу, який показує рівень участі учнів.

Ігор Гачков в роботі [11] розглядає пакет для «Булева алгебра» та «Теорія множин», які були спеціально розроблені для викладання курсу дискретної математики на основі підручника Раф П. Грімальді «Дискретна та конбінаторна математика в прикладному аспекті».

У статті [12] автор рекомендує калькулятор, який може обмежити булеві вирази (до чотирьох змінних) у вашому простому логічному виразі у формі Product Total (Sop). В роботі представлені функції процесора ARDUINO NANO. Схему вбудовано в друковану плату композитного формованого елемента.

Окрім наукових робіт було розглянуто також велику кількість програмних продуктів, які доступні у вільному в мережі Інтернет. Серед таких продуктів Boolean Algebra Solver [13], Boolean Expressions Calculator [14], Boolean Algebra Calculator [15], Online calculator for

Boolean functions [16], Boolean Algebra Simplifier Calculator [17], KSU Online [18] та інші.

Більшість із розглянутих програм та додатків це сервіси для онлайн розрахунків булевих функцій, або сервіси для мінімізації функцій за допомогою карт Карно. Недоліком таких програм є те, що користувач не отримує знань, а лише отримує відповідь, в кращому випадку з розв'язком. Такими калькуляторами зручно користуватись, коли потрібно перевірити результат, який користувач отримав після розрахунків, але такі калькулятори мають свої недоліки. До такого недоліку можна віднести недостатню точність – вони часто не забезпечують точність, яку можуть забезпечити стандартні калькулятори, що використовуються у реальному житті. Це може бути особливо проблематично при виконанні складних математичних розрахунків. Недостатня точність розрахунків може призвести до неправильного результату в роботі або в наукових дослідженнях. Також такі калькулятори можуть містити не всі необхідні функції.

Важливо, щоб користувачі мали можливість навчатись. Існують сервіси такі як KSU Online, MOODLE. Це програмні забезпечення для створення і управління електронними курсами (e-learning). За допомогою цих програмних продуктів можна створити персональні кабінети викладача та студента. Користувач може додати навчальні курси, наповнити їх методичними матеріалами, додати завдання, є форуми та інше. Окрім того, є вбудована система оцінювання та відстеження прогресу студентів, яка дозволяє вчителям аналізувати даних та відстежувати успішність учнів. Але недоліком даних програмних засобів є те, що задачі з дискретної математики там немає можливості оформити. В таблиці 1.2 представлено порівняльні характеристики програмних продуктів.

Таблиця 1.2

Програмний продукт	**Онлайн калькулятор**	**Персональний кабінет викладача та студента**	**Онлайн навчання та розміщення навчальних матеріалів**	**Оформлення завдань до модулів/уроків**
MiniBool	+	+	+	
KSU Online	-	+	+	-
MOODLE	-	+	+	-
Boolean Algebra Solver	+	-	-	-
Boolean Expressions Calculator	+	-	-	-
Boolean Algebra Calculator	+	-	-	-
Online calculator for Boolean functions	+	-	-	-
Boolean Algebra Simplifier Calculator	+	-	-	-

За допомогою програмного середовища навчального призначення для вивчення булевих функцій та схем функціональних елементів можна більш глибоко вивчити тему та набути більше практичних навичок. Аналізуючи таблицю 1.2 можна зробити висновок, що розробка програмного середовища залишається актуальною, тому що існуючі аналоги не охоплюють всіх потреб і вимог, які необхідні для вивчення дисципліни.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО СЕРЕДОВИЩА

2.1 Вимоги до програмного середовища

Вимоги до програмного середовища навчального призначення можна розділити на вимоги до функціональності, якості та інтерфейсу користувача.

Вимоги до функціональності програми визначають те, що програма повинна робити, які функції та операції вона повинна виконувати, і які результати вона повинна повертати. Вони визначають, які функції або можливості має мати програма, які операції вона повинна виконувати, які обмеження мають бути на її роботу та інше [19].

Програмний продукт має передбачати 3 типи користувачів, які мають різні можливості в системі і способи реєстрації (рисунок 2.1):

– адміністратор – користувач, який є в системі від моменту встановлення програмного продукту і без реєстрації;
– викладач – користувач, якого реєструє в системі адміністратор;
– студент – користувач, який має змогу самостійно зареєструватись в системі.

Рисунок 2.1 – Типи користувачів

Основними можливостями адміністратора мають бути підтримка та супровід програми, а також реєстрація нових користувачів, зокрема викладачів та їх видалення з системи.

Викладач не має можливості самостійно зареєструватись в системі, його профіль створюється адміністратором. Для того, щоб зареєструватись – викладач особисто має звернутись до адміністратора і він надасть викладачеві дані для входу в особистий кабінет. Викладач за отриманими даними для входу від адміністратора може авторизуватись в системі. В подальшому після реєстрації викладач буде мати можливість створити курс, який складається з декількох модулів. Для цих модулів має бути можливість додати навчальні тексти, прикріпити зображення, документ та посилання на інший ресурс. Також має бути можливість створити задачник до модуля. Основні чотири типи задач, які може створити викладач:

– знаходження булевої функції за таблицею істинності;
– побудова таблиці істинності за заданою булевою функцією;
– побудова функціональної схеми за булевою функцією;
– мінімізація булевої функції за допомогою карт Карно.

Викладачу має бути доступна таблиця зі студентами, які записані на його курс з наступними полями: ПІБ, група, виконані завдання за темами. Доступна функція видалення студентів з навчального курсу.

Студент – це єдиний тип користувача, який може самостійно створити акаунт за допомогою форми реєстрації. Після реєстрації студент може переглянути перелік навчальних курсів, підписатись і відписатись від навчального курсу. Після підписки на курс студент можна проходити задачі із задачника, після чого їх буде перевірено автоматично.

Вимоги до якості програми є критично важливими, оскільки якість впливає користувачів і результати функціонування програмного середовища. Програма повинна працювати вірно, виконувати всі запити користувача та давати очікуваний результат. Крім того вона повинна бути

стійкою до збоїв і помилок, та відновлюватися від них без втрати даних. Важливо також, щоб програма працювала швидко, забезпечуючи користувачам максимально можливу продуктивність і час відгуку.

Важливо щоб програмне середовище працювало у всіх сучасних веб-браузерах, забезпечуючи сумісність з усіма операційними системами та пристроями. Має бути забезпечений захист від вторгнень та зловмисних дій, забезпечуючи конфіденційність, цінність та доступність даних користувачів. Програма має бути здатною до масштабування, щоб забезпечити стабільну роботу при зростанні кількості користувачів та даних.

Основною вимогою до інтерфейсу є його простота. Програма має бути легкою у користуванні, зручною та інтуїтивно зрозумілою для користувачів.

2.2 Функціональні можливості програмного середовища

Функціональні можливості програмного середовища навчального призначення залежать від того, який тип користувача авторизовано в системі. Гості, тобто незареєстровані користувачі можуть створити власний обліковий запис, заповнивши форму реєстрації. Після такої реєстрації користувач може авторизуватись в системі як «Студент».

До основних функціональних можливостей можна віднести наступні:

– робота з обліковим записом – створення сторінки викладача, або реєстрація нового студента та авторизація і вхід до особистого кабінету;

– списки користувачів – виведення списків викладачів в кабінеті адміністратора та виведення студентів, записаних на навчальний курс викладача на сторінці викладача;

– організація навчальних курсів – створення навчальних курсів та наповнення їх інформацією (текстова інформація, зображення, файли та посилання);

– запис на навчальний курс – вибір навчального курсу студентом з подальшою підпискою на курс (або відпискою за необхідністю);

– задачник – створення задач викладачем курсу та можливість розв’язання їх студентом з автоматичною перевіркою;

– видалення інформації – можливість видалення користувачів, курсів, файлів і т.д.

2.3 Архітектура програмного середовища

Структура програмного середовища навчального призначення, яка включає програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними є його архітектурою. Опис архітектури програми є важливим етапом, тому що від неї залежить ціна на підтримку і розробку нового функціоналу, трудовитрати на побудову цілої системи з використанням даної архітектури.

Будь-який програмний код має взаємозалежності одного коду від іншого. Класи вимагають наявності нових класів, а функції викликають інші. Якщо залежностями неправильно управляти, то код стає складніше розуміти, він стає менш гнучким і виникає більше поломок.

Ознаками неякісної архітектури є наступні ознаки:

– програма чинить опір, невідомо скільки часу займе внесення змін в код, тому що вони впливають на інший функціонал;
– виникають баги в місцях, де зміни вносились раніше, або не вносились взагалі, а спричинено змінами в іншій частині коду;
– неможливо використовувати компоненти незалежно від інших компонентів;
– розмір проекту більший ніж міг би бути;

– надмірна складність, деякий функціонал приховує основну суть програми.

Для того щоб уникнути таких проблем і отримати якісний продукт, потрібно дотримуватись наступних принципів:

– зосереджувати код в одному місці, якщо він відповідає за якийсь один функціонал;
– має бути мінімальна залежність між класами;
– класи мають містити дані і методи для оперування цими даними, а не цікавитись даними з інших класів;
– класи мають бути відкриті для розширень, але закриті для модифікації.

Тому для того, щоб не отримати неякісну архітектуру і отримати якісний продукт – можна використовувати вже готові рішення, наприклад, framework-систему. Вона дає можливість розробника говорити про складні речі простіше і такі системи мають власні архітектурні стандарти.

Найдоцільніше використати архітектуру з поєднанням різних стилів. Перший стиль, який можна виділити – клієнт/сервер. Це стиль за якого програму умовно можна розділити на дві програми, одна відповідає за відображення даних, а інша за обробку даних сервером. Перша програма отримує дані у другої через запити. Також можна виділити компонентну архітектуру. Коли весь дизайн розбито на логічні функціональні компоненти. Кожен компонент буде виконувати певну функцію та може взаємодіяти з іншими компонентами програми.

2.4 Аналіз потреб користувачів та вимог до програми

Перш ніж перейти до визначення потреб та користувачів та їх вимог потрібно визначитись з цільовою аудиторією програмного середовища. Тобто потрібно визначитись хто буде користуватись програмою, як правило у цих людей спільні інтереси, потреби, теми.

Для того, щоб більш детально вивчити портрет потенційного користувача, потрібно дати відповідь на наступні запитання:

1. В якому регіоні проживають потенційні клієнти?
   Відповідь: географія неважлива, головне щоб був доступ до інтернету.
2. Який вік та стать клієнтів?
   Відповідь: переважна більшість студентів це чоловіки 65% та жінки 35% віком від 15 до 35 років.
3. Який рівень прибутку потенційних клієнтів?
   Відповідь: люди з будь-яким прибутком, програма не для продажу і не має платних підписок.
4. Яка освіта потенційних клієнтів?
   Відповідь: переважна більшість школярі старших класів, або студенти перших років навчання, близько 70%, 30% – люди з вищою освітою.

Таким чином, визначившись з цільовою аудиторією користувачів – можна описати історію користувачів. За допомогою програмного середовища студенти мають змогу з легкістю отримувати знання з булевої алгебри і закріплювати їх, виконуючи практичні задачі, в результаті чого отримають розрахований результат. Викладач в свою чергу має можливість додати навчальні матеріали та додати завдання для самоконтролю студентів.

Основною потребою користувачів є легкість у використанні та доступність, а головний критерій для студента – це якість отриманих знань, яка вже залежить не від самої програми, а від програми з викладачем в комплексі. Тому що навчає не програма, а навчає викладач, а програма лише засіб для кращого засвоєння матеріалу.

Прототип інтерфейсу можна представити в вигляді схематичного представлення блоків, що визначають функціонал програми. Приклад структури програми представлено на рисунку 2.2.

На схемі можна побачити як заме працює програма. Видно, що основний функціонал доступний після авторизації. А для того, щоб отримати доступи до персональних кабінетів потрібно зареєструватись, якщо користувач новий, або потрібно щоб користувача зареєстрував адміністратор, у випадку, якщо це викладач. Після авторизації користувач може виконати функції, які відповідають його блоку.

Для ілюстрації взаємодії користувача з і системою використовують UseCase-діаграму. На рисунку 2.3 представлено UseCase діаграму, події, які виконуються при взаємодії з програмою.

Рисунок 2.2 – Інформаційна структура програмного середовища

UseCase – методологія проектування програмного забезпечення, мета якої є визначення функціональних вимог для системи, на основі взаємодії користувачів з системою [20].

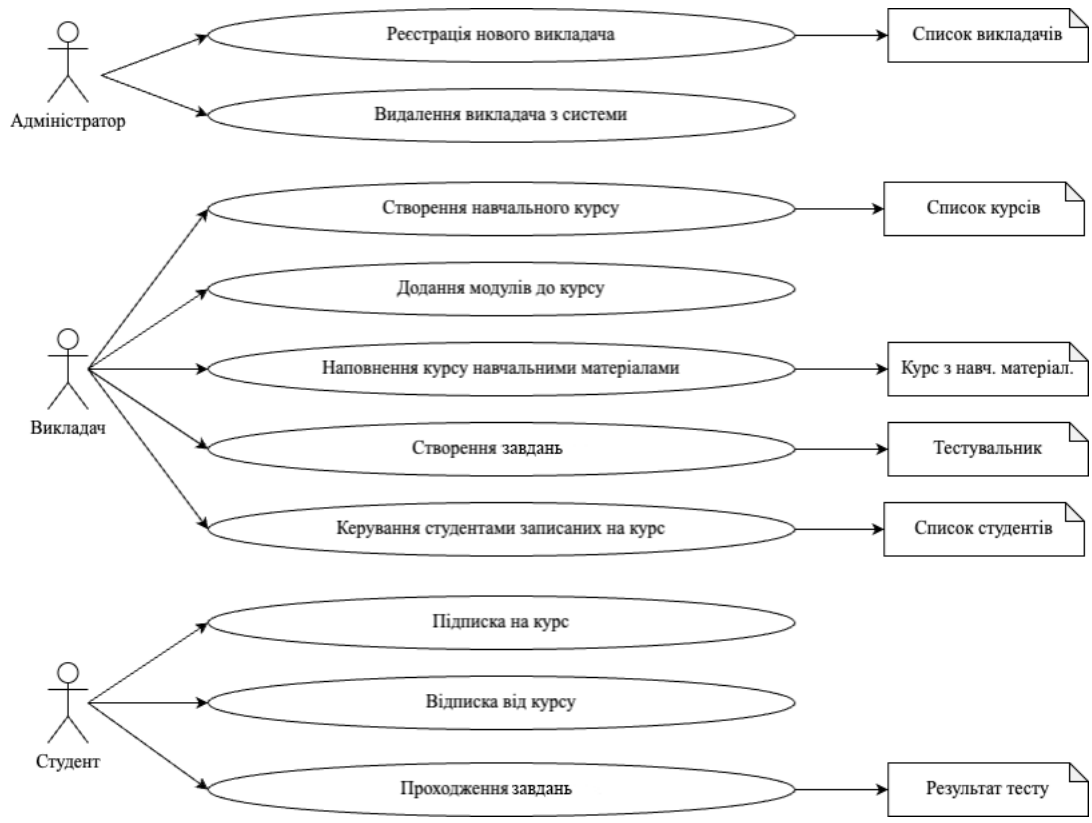

Рисунок 2.3 – UseCase-діаграма

Діаграма також допомагає визначити ролі користувачів, які будуть взаємодіяти з системою, і визначити різні сценарії, які можуть виникнути.

ПРЕЦИДЕНТ: РЕЄСТРАЦІЯ НОВОГО ВИКЛАДАЧА

Ектор: адміністратор.

Передумова: заявка від викладача на реєстрацію.

Післяумова: дані для входу в особистий кабінет викладача.

Сценарій:

1. Викладач звертається з проханням зареєструвати його в системі.

2. Адміністратор заповнює форму і додає інформацію про викладача в базу даних.

3. Викладач отримує дані для входу в систему.

ПРЕЦИДЕНТ: ВИДАЛЕННЯ ВИКЛАДАЧА З СИСТЕМИ

Ектор: адміністратор.

Передумова: заявка від викладача на видалення, або порушення правил.

Післяумова: дані для входу в систему викладача не дійсні.

Сценарій:

1. Адміністратор видаляє з бази викладача.

2. Викладач не може більше увійти в систему.

ПРЕЦИДЕНТ: СТВОРЕННЯ НАВЧАЛЬНОГО КУРСУ

Ектор: викладач.

Передумова: є недоданий курс.

Післяумова: курс додано і доступний для студентів.

Сценарій:

1. Викладач заповнює форму з даними про навчальний курс.

2. Викладач опубліковує курс в загальний доступ.

ПРЕЦИДЕНТ: ДОДАННЯ МОДУЛІВ ДО КУРСУ

Ектор: викладач.

Передумова: є курс без модулів.

Післяумова: модулі додано до курсу.

Сценарій:

1. Викладач заповнює форму з даними про модулі навчального курсу.

2. Викладач опубліковує заповнену інформацію.

ПРЕЦИДЕНТ: НАПОВНЕННЯ КУРСУ НАВЧАЛЬНИМИ МАТЕРІАЛАМИ

Ектор: викладач.

Передумова: є курс без доданого навчального матеріалу.

Післяумова: навчальні матеріали додано до курсу.

Сценарій:

1. Викладач додає текст, зображення, файл, посилання до навчального курсу.

2. Викладач опубліковує навчальні матеріали до курсу.

ПРЕЦИДЕНТ: СТВОРЕННЯ ЗАДАЧНИКА

Ектор: викладач.

Передумова: додання практичних завдань.

Післяумова: створено завдання для самоконтролю.

Сценарій:

1. Викладач обирає тип задач.

2. Заповнення умови завдання.

3. Публікація задач.

ПРЕЦИДЕНТ: КЕРУВАННЯ СТУДЕНТАМИ ЗАПИСАНИХ НА КУРС

Ектор: викладач.

Післяумова: оновлений список студентів.

Сценарій:

1. Викладач відкриває список студентів навчального курсу.

2. Видалення студента, якщо це необхідно.

3. Аналіз успішності студента.

ПРЕЦИДЕНТ: ПІДПИСКА НА КУРС

Ектор: студент.

Передумова: сподобався навчальний курс.

Післяумова: доступний для вивчення курс.

Сценарій:

1. Студент переглядає список курсів.

2. Знайомиться з інформацією про навчальний курс.

3. Підписка на навчальний курс.

ПРЕЦИДЕНТ: ВІДПИСКА ВІД КУРСУ

Ектор: студент.

Передумова: навчальний курс зайвий в списку.

Післяумова: відписався.

Сценарій:

1. Студент відписується від навчального курсу.

ПРЕЦИДЕНТ: ПРОХОДЖЕННЯ ЗАВДАНЬ

Ектор: студент.

Передумова: не пройдено завдання.

Післяумова: отримано результат.

Сценарій:

1. Студент обирає задачу для проходження.
2. Розв’язує її.
3. Отримує автоматично розрахований результат.

РОЗДІЛ 3
ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ПРОГРАМИ

В попередньому розділі дипломної роботи було визначено вимоги, функціонал та архітектуру програмного середовища навчального призначення. Маючи цю інформацію, перед початком розробки програми потрібно визначитись з набором інструментів, методів та підходів, які будуть використовуватись при розробці. Ці технології допоможуть забезпечити ефективнішу роботу розробника, зменшуючи час, затрачуваний на розробку, тестування та розгортання програм.

До технологій для розробки програмного середовища входять мови програмування, фреймворки, бібліотеки, інструменти для керування версіями, тестування та розгортання програм, інтеграції та постачання, сервіси хмарної інфраструктури, віртуалізації, контейнеризації та інші технології.

В якості основного фреймворку для розробки frontend (розробка клієнтської частини програмного середовища) та backend (розробка серверної частини програмного середовища) буде використовуватись Next.js [21]. Даний фреймворк базується на Node.js та React. Він дозволяє створювати веб-додатки з покращеною продуктивністю та покращеним користувацьким досвідом за допомогою покращених функцій попереднього рендерингу, а саме рендерингу на стороні серверу – SSR та генерація статичних сторінок – SSG. Тобто додатки Next.js використовують всі переваги React і просто додають декілька нових функцій.

SSR дає можливість сторінкам завантажуватись швидше і покращує зручність роботи з програмою, через покращений відгук, схема роботи SSR представлена на рисунку 3.1. Використання рендерингу на стороні серверу дає переваги в SEO, що дає більш високі позиції програми при пошуку в мережі інтернет. Перевагою є те, що всі компоненти крім

клієнта надсилають свою інформацію на сервер і клієнт може отримати попередню версію HTML-сторінку, замість того, щоб кожен раз запитувати доступ до компонентів окремо і рендерити їх на стороні клієнта.

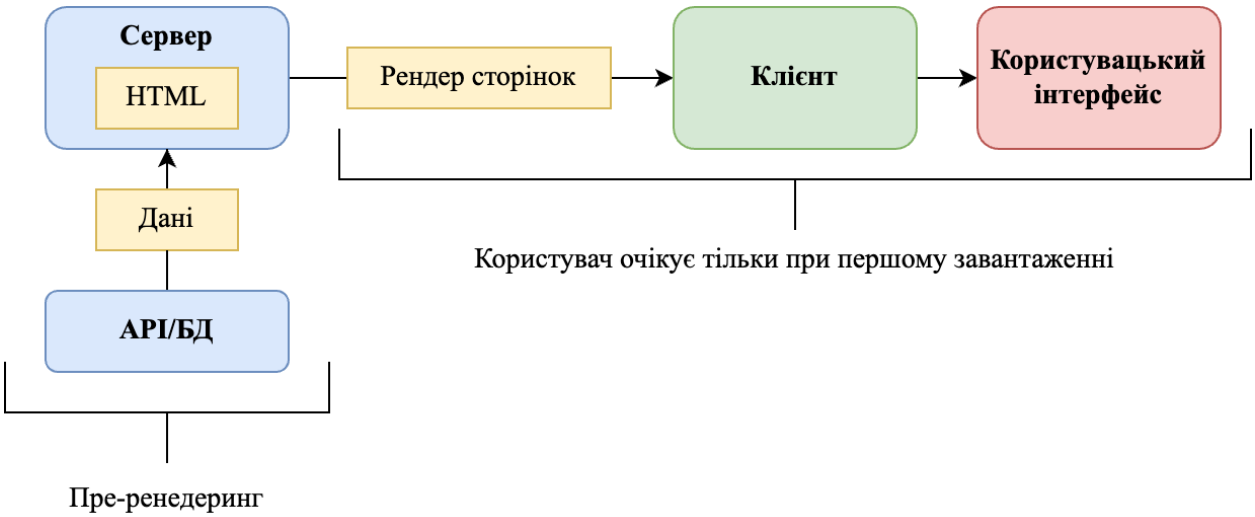

Рисунок 3.1 – Принцип роботи SSR

Статичні сторінки генеруються на етапі збірки, а не на сервері при запиті користувача. Це дозволяє зменшити навантаження на сервер та збільшити швидкість завантаження сторінок завдяки SSG. Next.js дозволяє швидко та легко налаштувати веб-додатки за допомогою конфігураційних файлів та плагінів. Так-як фреймворк побудовано на базі React, то це дозволяє розробникам використовувати React-компоненти для побудови програм. Таким чином Next.js можна використовувати як для клієнтської так і для серверної частини програми, що забезпечує клієнт-серверну архітектуру, яку було запропоновано в підрозділі 2.3.

Для збереження, організації та отримання даних про навчальні курси, користувачів, завдання, задачі та інше – потрібно використовувати базу даних. Її основна мета – забезпечити ефективний доступ до даних та зберігати їх в безпечному місці. Застосування баз даних дозволяє програмі зберігати значно більше інформації, ніж може бути збережено у

файлі, і зручно її обробляти. В якості такої бази даних буде використовуватись SQLite [22]. SQLite – це вбудована кросплатформна БД, яка підтримує досить повний набір команд SQL і доступна у вихідних кодах (мовою С).

Перевагами використання такої бази даних є:

– ефективний доступ до даних – вона дозволяє швидко знаходити та отримувати інформацію, що забезпечує більш ефективну роботу програми;
– збереження великої кількості інформації – вона дозволяє зберігати значно більше інформації, ніж може бути збережено у файлі;
– легке збереження та оновлення даних – можна легко зберігати та оновлювати інформацію, що дозволяє швидко вносити зміни у програму;
– забезпечення безпеки даних – дозволяє зберігати дані в безпечному місці, що зменшує ризик їх втрати чи пошкодження;
– інтеграція з іншими програмами – дозволяє легко обмінюватися даними між різними програмами, що забезпечує більшу ефективність та зручність роботи з даними;
– забезпечення консистентності даних – забезпечує консистентність даних та уникнення дублювання інформації.

У загальному, використання SQLite є дуже важливою складовою розробки програмного середовища, що дозволяє зберігати, організовувати та отримувати доступ до великої кількості даних в ефективний та безпечний спосіб.

Для того щоб працювати з базою даних за допомогою JavaScript без використання SQL використовують ORM інструменти. Таким інструментом є Prisma – це сучасне об'єктно-реляційне відображення [23]. Робота з базами даних – одна з найскладніших задач, що виникає при

розробці програм. Prisma пропонує рішення, що дає змогу зосередитися на даних замість SQL.

Робота починається з визначення схеми. Схема дає змогу визначати моделі за допомогою інтуїтивно зрозумілої мови. Вона також містить з'єднання з БД і визначає генератор. Кожна модель прив'язана до таблиці в БД і є основою для генерованого Prisma Client інтерфейсу доступу до даних.

Prisma Client дає змогу розробникам мислити в категоріях об'єктів. Іншими словами, замість концепції екземплярів моделі, у відповідь на запит до БД повертаються звичайні JavaScript-об'єкти. Крім того, запити є повністю типізованими. На рисунку 3.2 представлено схему роботи ORM.

Рисунок 3.2 – Схема роботи Prisma ORM

Авторизація буде здійснюватися на стороні сервера за допомогою NextAuth [24] з використанням CSRF Cookies, а також сесій на стороні сервера. NextAuth.js – це повне рішення для автентифікації з відкритим вихідним кодом для додатків Next.js. Воно розроблене з нуля для підтримки Next.js та Serverless.

NextAuth містить наступні пакети/проекти:

– Основний пакет next-auth;
– Тестовий додаток для розробки;

– Всі пакети @next-auth/*-adapter;
– Сайт з документацією.

Запити до API будуть здійснюватися за допомогою React Query. React Query – популярна бібліотека для керування та кешування стану сервера в React-додатках. Вона надає набір хуків та утиліт для отримання, кешування та оновлення даних з API, а також для вирішення загальних проблем, таких як кешування, фонові оновлення та обробка помилок.

Однією з ключових особливостей React Query є його здатність автоматично кешувати дані та оновлювати їх у фоновому режимі, завдяки чому компоненти інтерфейсу завжди можуть відображати найсвіжіші дані без необхідності керувати станом вручну. Він також надає ряд вбудованих функцій для обробки оптимістичних оновлень, опитувань та пагінації.

React Query підтримує різні типи методів отримання даних, такі як REST API, GraphQL і навіть веб-сокети. Він також має вбудовану підтримку для роботи з автентифікацією та авторизацією, що може бути поширеною проблемою в багатьох додатках.

Загалом, React Query може значно спростити процес отримання даних та управління станом сервера в React-додатках, і є популярним вибором серед багатьох React-розробників.

Для відображення блок-схем буде використовуватись React Flow. React Flow – це легка бібліотека для побудови інтерактивних графічних інтерфейсів користувача на основі вузлів у React-додатках. Вона надає набір компонентів та утиліт для створення візуальних робочих процесів, діаграм та блок-схем, які можна використовувати для представлення складних систем або процесів.

React Flow розроблений таким чином, щоб бути гнучким та легко налаштовуваним, і підтримує широкий спектр варіантів кастомізації своїх вузлів, ребер та інших компонентів інтерфейсу користувача. Він також надає набір вбудованих обробників подій та зворотних викликів, які

можна використовувати для реагування на взаємодію з користувачем та оновлення інтерфейсу в режимі реального часу.

Однією з ключових особливостей React Flow є підтримка різних типів графіків та макетів, які можна використовувати для представлення різних типів даних та зв'язків. Він також надає ряд вбудованих плагінів і розширень, таких як можливість масштабування і панорамування графіка, або перетягування вузлів для їхньої зміни.

Загалом, це потужна та гнучка бібліотека для побудови інтерактивних графічних інтерфейсів користувача в React-додатках, яка може бути використана для широкого спектру застосувань, від візуалізації складних даних до побудови кастомних робочих циклів та процесів.

Для створення верстки буде використовуватись фреймворк TailwindCSS. TailwindCSS – це CSS-фреймворк, який надає набір попередньо визначених класів CSS, які можна використовувати для швидкого створення користувацьких інтерфейсів у веб-додатках. На відміну від інших CSS-фреймворків, які надають готові компоненти інтерфейсу, TailwindCSS фокусується на наданні набору низькорівневих утилітарних класів, які можна використовувати для створення користувацьких компонентів.

Філософія TailwindCSS полягає в тому, щоб надати розробникам можливість швидко і легко створювати користувацькі інтерфейси без необхідності писати багато власного CSS-коду. Це досягається шляхом надання набору попередньо визначених класів CSS, які можна комбінувати для досягнення широкого спектру візуальних ефектів.

Деякі з ключових особливостей TailwindCSS включають:

– Широкий набір класів утиліт: TailwindCSS надає широкий спектр попередньо визначених класів утиліт, які можна використовувати для швидкого стилювання елементів HTML.

– Настроюється конфігурація: Розробники можуть налаштовувати конфігураційний файл TailwindCSS, щоб додавати або видаляти класи, налаштовувати кольори і типографіку і багато іншого.
– Адаптивний дизайн: TailwindCSS включає вбудовану підтримку адаптивного дизайну, що дозволяє розробникам легко створювати адаптивні інтерфейси, які адаптуються до різних розмірів екрану.
– Низька специфічність: Використовуючи утилітарні класи, TailwindCSS уникає проблеми надмірно специфічних CSS-селекторів, які можуть ускладнити перевизначення стилів.

Загалом, TailwindCSS є популярним вибором серед веб-розробників, які цінують гнучкість і продуктивність, і він був широко прийнятий багатьма популярними веб-додатками і фреймворками.

Таким чином було визначено стек технологій, які буде використано при розробці програмного середовища навчального призначення для вивчення булевих функцій і схем з функціональними елементами. Обрані технології забезпечать якісну архітектуру, тому функціонал можна буде розширювати без проблем, а також забезпечить високу швидкість роботи.

РОЗДІЛ 4
РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Розробка моделі бази даних

Розробка моделі бази даних – це процес створення структури для зберігання даних в базі даних. У процесі розробки моделі бази даних необхідно враховувати потреби користувачів та бізнес-вимоги. Для цього можна провести аналіз вимог та взаємодії з потенційними користувачами бази даних. Також потрібно враховувати фактори безпеки та забезпечення конфіденційності даних, а також дотримуватися принципів нормалізації бази даних для забезпечення її ефективності та цілісності. У процесі розробки моделі бази даних можна використовувати різні методи та інструменти, такі як UML-діаграми, ER-діаграми, реляційні моделі даних та інші.

В процесі розробки моделі бази даних було пройдено наступні етапи:

1. Визначення вимог до бази даних: на цьому етапі було визначено, які дані потрібно зберігати в базі даних, які операції будуть виконуватися з даними.
2. Створення концептуальної моделі: на цьому етапі було створено модель, яка описує сутності та зв'язки між ними в базі даних.
3. Створення логічної моделі: на цьому етапі було створено модель, яка описує структуру даних в базі даних з точки зору програмного забезпечення.
4. Створення фізичної моделі: на цьому етапі було визначено, як будуть зберігатися дані в базі даних на рівні операційної системи. Це може включати в себе визначення типів даних, структури таблиць та індексів.

5. Розробка бази даних: на цьому етапі була створена сама база даних за допомогою вибраної СУБД та створеного на попередніх етапах моделювання.
6. Тестування та відладка: на цьому етапі базу даних було перевірено на працездатність, а також виконано дії з заповнення тестовими даними.

В якості бази даних обрано SQLite. Це вбудовувана реляційна база даних, яка зберігає дані в локальному файлі. Вона не потребує окремого сервера баз даних, оскільки її можна використовувати в рамках додатків, що використовують її як бібліотеку. Вона підтримує стандартні функції баз даних, такі як SQL-запити, транзакції, індексування та забезпечення цілісності даних. Вона є досить легкою в налаштуванні та використанні і може бути використана в багатьох різних типах програм, від мобільних додатків до великих серверних додатків. Однією з переваг бази даних SQLite є її швидкодія, оскільки база даних зберігається в локальному файлі, що дозволяє швидко зчитувати дані з неї.

Інформація програмного середовища зберігається в семи таблицях.

Таблиця «User» містить наступні поля: ідентифікатор, електронна пошта, пароль, ім’я, прізвище, крупа, роль, курс, підписка і відповіді на завдання.

Таблиця «Course» містить наступні поля: ідентифікатор, назва, опис, дата створення, модулі, прикріпленні файли, ідентифікатор користувача, користувач, підписники та завдання.

Таблиця «Challenge» містить наступні поля: ідентифікатор, назва, тіло, дата створення, ідентифікатор курсу, курс, відповіді.

Таблиця «ChallengeAnswer» містить наступні поля: ідентифікатор завдання, ідентифікатор, ідентифікатор користувача, користувач, статус, дата створення.

Таблиця «UserSubscription» містить наступні поля: ідентифікатор користувача, ідентифікатор курсу, користувач та курс.

Таблиця «Module» містить наступні поля: ідентифікатор, назва, опис, ідентифікатор курсу, курс, дата створення.

Таблиця «Attachment» містить наступні поля: назва файлу, ім’я збереженого файла, ідентифікатор курсу, курс.

На рисунку 4.1 представлено повну модель даних, що відображає взаємозв'язок між таблицями.

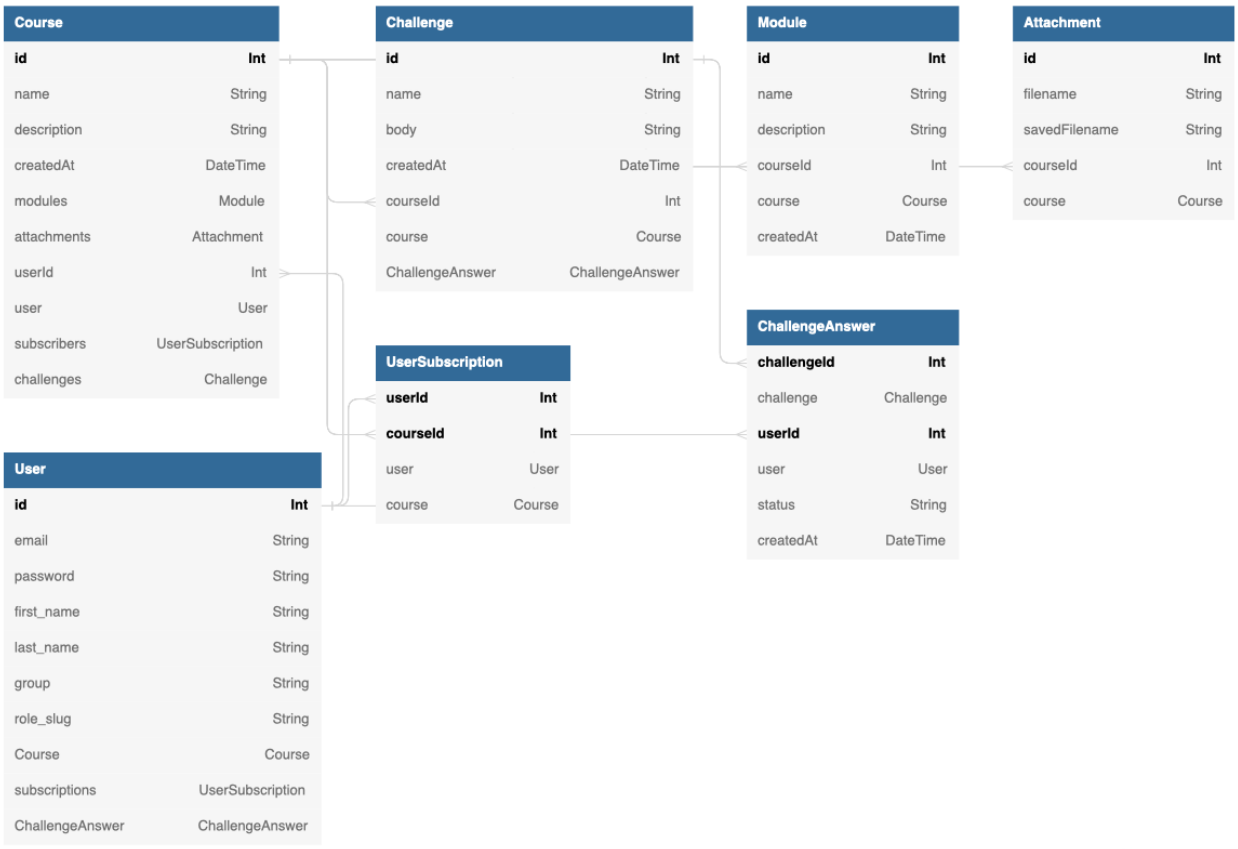

Рисунок 4.1 – ER-діаграма

4.2 Верстка користувацьких інтерфейсів

Створення користувацьких інтерфейсів – це frontend розробка. Цей процес включає в себе розробку веб-сторінок, взаємодію з користувачем та інші елементи, які є важливими для забезпечення задоволення користувачів. Frontend розробка включає в себе всі етапи від

розробки дизайну до написання коду, тестування та розгортання веб-інтерфейсів.

Як було зазначено раніше, для розробки інтерфейсів використовується Next.js. Він надає багато інструментів для розробки користувацьких інтерфейсів. Next.js використовує React для рендерингу компонентів, які можуть бути використані для створення різних елементів інтерфейсу, таких як форми, кнопки, списки тощо.

В роботі було створено і динамічні сторінки, які можуть змінюватися в залежності від даних, що надходять з сервера. Це дозволяє створювати інтерактивні інтерфейси, які реагують на дії користувача та надають їм більше можливостей.

Для стилізації інтерфейсу, використано Tailwind, який дозволяє створювати індивідуальні стилі для кожного компонента, не змінюючи при цьому інших елементів інтерфейсу. Це дозволило забезпечити консистентний вигляд інтерфейсу та зручну розробку.

Next.js використовує компонент App для ініціалізації сторінок, в ньому відбувається:

– збереження макетів між змінами сторінок;
– збереження стану під час навігації сторінками;
– додавання глобальних CSS.

Next.js використовує компонент Document – це спеціальний компонент, який використовується для розширення тегів html і body програмного середовища. Це необхідно, тому що сторінки Next.js пропускають певну розмітку середовища документа.

В директорії «pages» знаходяться сторінки – це компоненти, які експортуються з файлів з розширенням .tsx. Кожна сторінка асоціюється з маршрутом за назвою. Наприклад, сторінка pages/create-course.tsx буде доступна за адресом https://domain.com/create-course, де domain.com – це домен веб-додатку.

Маршрут для сторінки pages/course[id].tsx буде динамічним, тобто така сторінка буде доступною за адресом https://domain.com/course/n, де «n» буде змінюватись в залежності від сторінки.

За замовчуванням усі сторінки рендеряться попередньо (pre-rendering). Це призводить до кращої продуктивності та SEO. Кожна сторінка асоціюється з мінімальною кількістю JavaScript. Під час завантаження сторінки запускається JavaScript -код, який робить її інтерактивною.

На деяких етапах життєвого циклу програмного середовища Next.js необхідні якісь файли. Це можуть бути іконки, власні шрифти, зображення тощо. У Next.js це називається «обслуговування статичних файлів№ і є єдине джерело для них – папка «public».

Загалом, розробка користувацьких інтерфейсів за допомогою Next.js є досить простою і ефективною, завдяки вбудованому підтримки React, CSS модулів і серверного рендерингу.

4.3 Розробка серверної частини програмного середовища

За допомогою Next.js можна використовувати будь-яку базу даних, в нашому випадку використовується SQLite. При роботі з базами даних використовується ORM для моделювання структури даних бази даних. В програмному середовищі для вивчення булевих функцій та схем функціональних елементів використовується Prisma як ORM, щоб керувати базою даних без зайвого клопоту.

Використовуючи Prisma було встановлено Prisma Client для нашого проєкту. Основний конфігураційний файл Prisma, який містить конфігурацію бази даних – «prisma/schema.prisma». Було створено моделі. Моделі представляють сутності предметної області програми. Вона описує, як дані будуть представлені у базі даних. Prisma використовує модель для створення бази даних, таблиць і всіх необхідних полів.

Для зв'язку із сервером у Next.js використовуються функції API (API Routes). Вони дають змогу створювати API-маршрути на сервері, які можна викликати з клієнтського коду на стороні фронтенду. Функції API створюються в спеціальній папці «pages/api».

Далі цю функцію API можна викликати за допомогою HTTP запиту з клієнтської частини. Next.js надає широкі можливості для інтеграції з backend-серверами та створення API-маршрутів на сервері.

Було встановлено NextAuth.js для авторизації користувачів. Було створено файл «pages/api/auth/[...nextauth].js» для конфігурації NextAuth.js і налаштування провайдерів аутентифікації. NextAuth.js полегшує інтеграцію автентифікації у нашому програмному середовищі і забезпечує безпеку застосунку за допомогою захисту від атаки CSRF та інших заходів безпеки.

4.4 Огляд та тестування програмного середовища

Програмне середовище навчального призначення для вивчення булевих функцій та схем функціональних елементів було виконано відповідно до поставленого завдання. Для початку роботи користувача – йому потрібно авторизуватися в системі. Приклад форми авторизації представлено на рисунку 4.2.

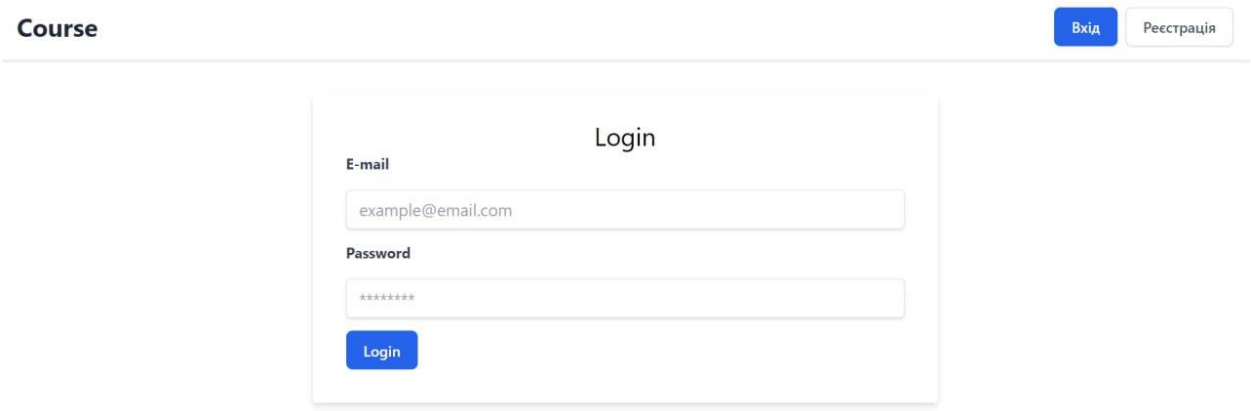

Рисунок 4.2 – Сторінка авторизації

Адміністратор сайту є в системі за замовчуванням і він має можливість додавати нових викладачів в систему. Приклад форми додавання нового викладача представлено на рисунку 4.3.

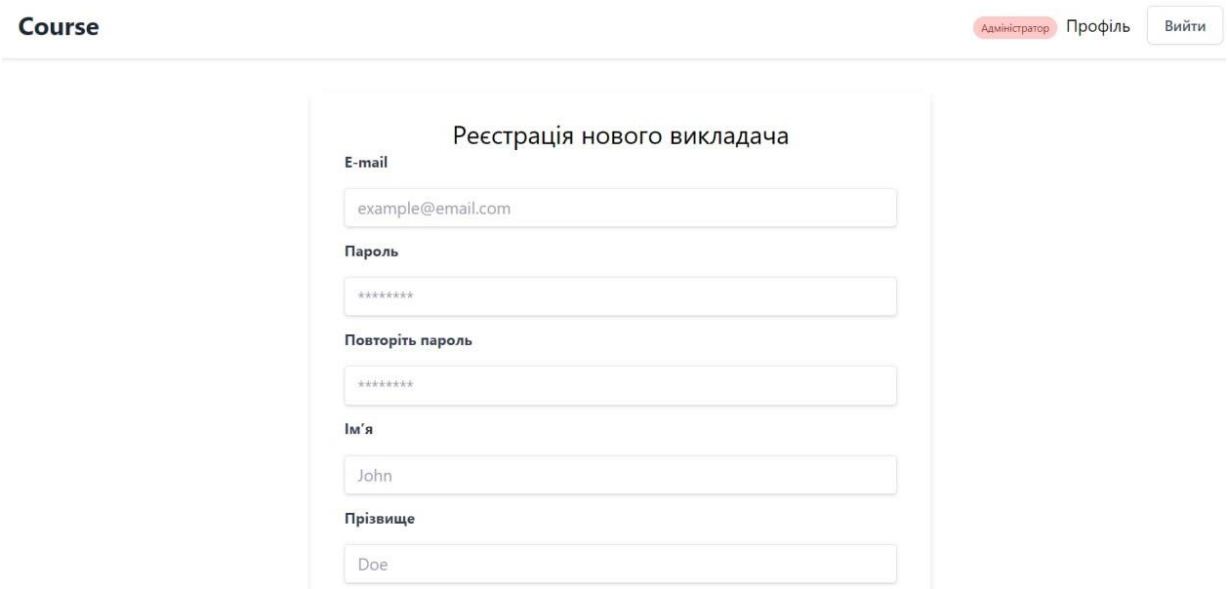

Рисунок 4.3 – Форма реєстрації нового викладача

Студент може самостійно зареєструватись в системі. Приклад сторінки реєстрації студента представлено на рисунку 4.4.

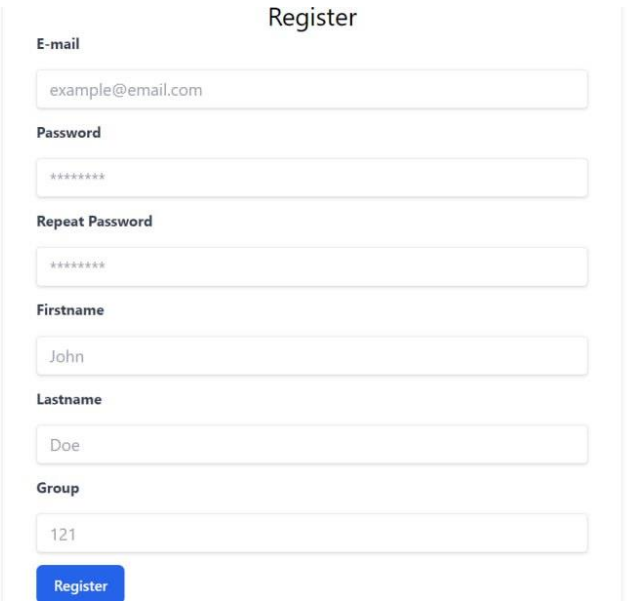

Рисунок 4.4 – Форма реєстрації нового студента

Сторінка з викладачами в кабінеті адміністратора показана на рисунку 4.5. Адміністратор може додати нового викладача, натиснувши «Додати викладача» і потрапить на форму, що показано на рисунку 4.3, а може видалити його, натиснувши на «Видалити» біля відповідного викладача.

Рисунок 4.5 – Список викладачів

Викладач може створювати нові курси, модулі та завдання. Для створення нового курсу потрібно натиснути на кнопку «Створити курс». На рисунку 4.6 показано сторінку з курсами викладача.

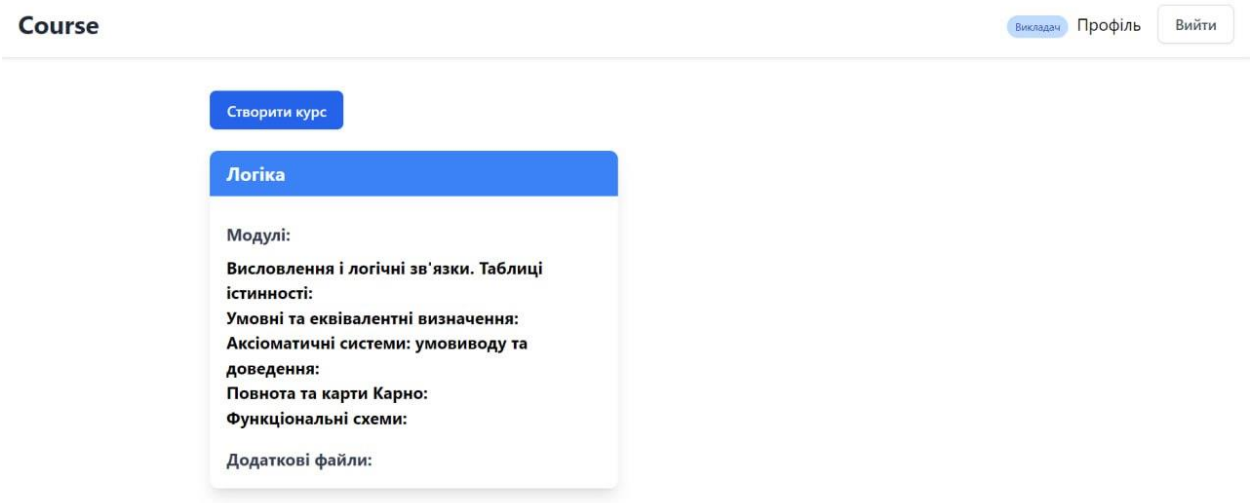

Рисунок 4.6 – Сторінка з курсами

Приклад форми створення курсу представлено на рисунку 4.7. Потрібно заповнити назву та опис курсу, а також можна додати файли та модуль навчального курсу.

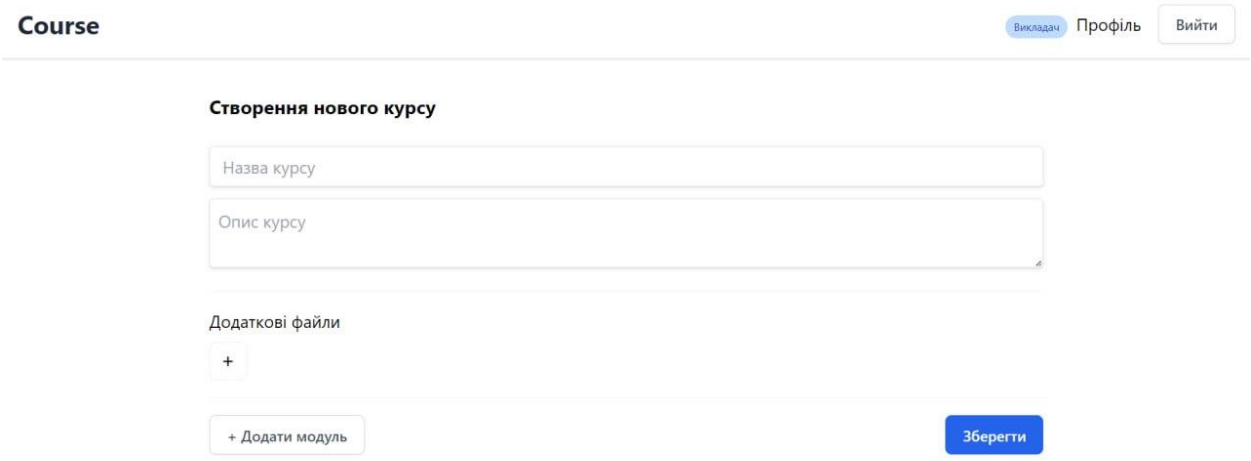

Рисунок 4.7 – Створення нового курсу викладачем

Далі студент обирає курс, який його цікавить і проходить навчання. Він має можливість пройти завдання, які залишив викладач і це завдання автоматично буде перевірено системою.

Для запуску програмного середовища необхідно виконати наступні кроки:

5. встановити NodeJS (версія 18.15 або вище) ;
6. встановити менеджер пакетів Yarn командою «npm install -g yarn»;
7. в консолі відкрити паку з проектом;
8. виконати послідовно команди: «yarn install» та «yarn db:setup»
9. для старту проекту виконати послідовно команди: «yarn build» (виконувати тільки після змін у файлах проекту) та «yarn start» (запускає проект)
10. після виконання команди yarn start, буде доступна веб-сторінка.

Для програмного середовища було проведено тестування інтерфейсу, функціональності і безпеки. Підсумовуючи, можна сказати, що програмне середовище має високу швидкість роботи. Помилок в роботі функціоналу виявлено не було, виконані перевірки введення даних в поля форми.

ВИСНОВКИ

З метою підвищення ефективності навчання та забезпечення якісної підготовки фахівців, розробка програмного середовища навчального призначення є важливим і актуальним завданням у сфері освіти. У ході дослідження було розглянуто теоретичні аспекти розробки програмного забезпечення для навчання, проаналізовано існуючі засоби, їх переваги та недоліки.

Для розробки програмного середовища було використано сучасні технології програмування та баз даних, з метою забезпечення зручного та швидкого доступу до навчального матеріалу, інтерактивного навчання та контролю знань студентів.

Розроблене програмне середовище забезпечує ефективне та зручне навчання студентів, дозволяє забезпечити індивідуальний підхід до кожного студента, а також зменшує час на оцінювання робіт та інші рутинні процеси.

Застосування даного програмного середовища може допомогти покращити процес навчання, зробити його більш доступним та зрозумілим для студентів. Крім того, таке рішення може забезпечити викладачів зручним та ефективним інструментом для розробки та контролю навчального процесу.

Отже, розробка програмного середовища навчального призначення для вивчення булевих функцій та схем функціональних елементів є важливим кроком у поліпшенні якості освіти, що дозволяє забезпечити ефективну та інтерактивну форму навчання, а також забезпечує зручний та швидкий доступ до навчального матеріалу для студентів та викладачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sheldon B. A. On a Theory of Boolean Functions // Journal of the Society for Industrial and Applied Mathematics. 1959., No. 4 Vol. 7. P. 487-498. DOI: https://doi.org/10.1137/0107041
2. Трохимчук Р. М., Нікітченко М. С. Дискретна математика у прикладах і задачах //URL: http://csc. knu. ua/media/filer_public/89/10/89101127-5400-4d61-9840-7eab32caddab/discrete_mathematics. pdf. – 2017.
3. Копча-Горячкіна Г.Е., Василенко Ю.А. Основи дискретної математики // Методичний посібник, частина II, 2006. – 59 с.
4. Побудова таблиці істинності й нормальної диз'юнктивної форми булевої функції // Київські учнівські олімпіади з інформаційних технологій і вивчення інформатики: [Веб-сайт]. URL: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2012/normalform/normalform.html (дата звернення: 01.03.2023).
5. Kearns M., Li M., Valiant L. Learning boolean formulas //Journal of the ACM (JACM). – 1994. – Т. 41. – №. 6. – С. 1298-1328.
6. Кичак В. М., Стронський В. В., Задорожний В. К. СИНТЕЗ ЦИФРОВИХ ПРИСТРОЇВ З П’ЯТЬМА ТА БІЛЬШЕ ВХОДАМИ НА ЛОГІЧНИХ ЕЛЕМЕНТАХ З ФУНКЦІОНАЛЬНИМ НАДЛИШКОМ //Математична модель передатної характеристики двотактного підсилювача постійного струму із симетричною структурою. – С. 26.
7. Мартіросян М. К., Князєв І. А. Аналіз методів Квайна і карт Карно мінімізації булевих функцій.
8. Jonnson B. , Tarski A. Boolean Algebras with Operators // American Journal of Mathematics. 1952., No. 1 Vol. 74. P. 127-162. DOI: https://doi.org/10.2307/2372074

9. Стасюк Ю. В., Парфьонова Т. О. Про розробку тренажера для дистанційного навчального курсу" Дискретна математика" з обчислення булевих функцій. – 2017.

10. Jiménez-Hernández E. M. et al. Using web-based gamified software to learn Boolean algebra simplification in a blended learning setting //Computer Applications in Engineering Education. – 2020. – Т. 28. – №. 6. – С. 1591-1611. DOI: https://doi.org/10.1002/cae.22335

11. GACHKOV I. Discrete Mathematics: Boolean algebra and Set theory with TI-83/89. – P. 1-14

12. Dhanush K., Ramesh K. B. Design and Development of Boolean Logic Simplifying Calculator //Advancement and Research in Instrumentation Engineering. – 2022. – Т. 4. – №. 3.

13. Boolean Algebra Solver - Boolean Expression Calculator: [Website]. URL: https://www.boolean-algebra.com/ (viewed on: 08.03.2023).

14. Boolean Expressions Calculator // DCode: [Website]. URL: https://www.dcode.fr/boolean-expressions-calculator (viewed on: 09.03.2023).

15. Boolean Algebra Calculator // EMathHelp: [Website]. URL: https://www.emathhelp.net/calculators/discrete-mathematics/boolean-algebra-calculator/ (viewed on: 10.03.2023).

16. Online calculator for Boolean functions: [Website]. URL: https://www.ii.uib.no/~mohamedaa/odbf/check.html (viewed on: 11.03.2023).

17. Boolean Algebra Simplifier Calculator: [Website]. URL: https://boolean-simplifier.com/ (viewed on: 12.03.2023).

18. KSU Online: [Веб-сайт]. URL: https://ksuonline.kspu.edu/ (дата звернення: 15.03.2023).

19. Супруненко О. Аналіз вимог до програмного забезпечення: навчальний матеріал. Черкаси: ЧНУ ім. Богдана Хмельницького, 2012. 24 с.

20. Fauzan R. et al. Use case diagram similarity measurement: A new approach //2019 12th International Conference on Information & Communication Technology and System (ICTS). – IEEE, 2019. – С. 3-7. DOI: https://doi.org/10.1109/ICTS.2019.8850978

21. Next.js by Vercel - The React Framework: [Website]. URL: https://nextjs.org/ (viewed on: 14.03.2023).

22. SQLite: [Website]. URL: https://sqlite.org/index.html (viewed on: 15.03.2023).

23. SQLite database connector // Prisma Data, Inc: [Website]. URL: https://www.prisma.io/docs/concepts/database-connectors/sqlite (viewed on: 15.03.2023).

24. NextAuth.js: [Website]. URL: https://next-auth.js.org/ (viewed on: 15.03.2023).

ДОДАТКИ

ДОДАТОК А. КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Руда Юліана Юріївна, учасниця освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

- дотримуватися:
  - вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
  - принципів та правил академічної доброчесності;
  - нульової толерантності до академічного плагіату;
  - моральних норм та правил етичної поведінки;
  - толерантного ставлення до інших;
  - дотримуватися високого рівня культури спілкування;
- надавати згоду на:
  - безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
  - оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
  - використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності,

використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного

іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов’язків, вчасно та сумлінно виконувати необхідні навчальні та науководослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв’язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов’язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов’язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров’ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

12.09.2019		Юліана Руда
(дата)	(підпис)	(ім’я, прізвище)