Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка навчальної системи для вивчення основ вебпрограмування»

Виконав: студент IV курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Дударко С.Є.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Войтко В. В.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Колесник І.С
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Дударку Сергію Євгеновичу

1. Тема роботи – «Розробка навчальної системи для вивчення основ вебпрограмування»

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. №80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: середовище розробки WebStorm, мова розробки JavaScript, фреймворк Vue, базові алгоритми.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку вебсистем для вивчення основ вебпрограмування; розробка методу, моделі та алгоритмів роботи системи; розробка програмних модулів; тестування навчальної системи; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів, метод додавання та редагування курсів із авторизацією, блок-схема алгоритму методу додавання та

редагування курсів із авторизацією, блок-схема загального алгоритму ро навчальної системи, розробка програми, тестування, висновки, апроб результатів роботи і публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завданн прийня
1-4	Войтко В.В., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку вебсистем для вивчення основ вебпрограмування	14.03.24 – 20.03.24	виконано
2	Розробка методів, моделі та алгоритмів роботи програми	22.03.24 – 12.04.24	виконано
3	Розробка програмних модулів	13.04.24 – 13.05.24	виконано
4	Тестування навчальної системи	14.05.24 – 20.05.24	виконано
5	Оформлення матеріалів до захисту БДР	21.05.24 – 30.05.24	виконано

Студент _____ (підпис) Дударко С.Є. (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) Войтко В.В. (прізвище та ініціали)

Анотація

Бакалаврська дипломна робота складається з 58 сторінок формату А4, на яких є 34 рисунки, 3 таблиці, список використаних джерел містить 21 найменування.

У бакалаврській дипломній роботі проведено аналіз стану навчальних систем для вивчення основ вебпрограмування. Описано об’єкт, предмет, завдання та методи дослідження. Сформульовано мету досліджень – підвищення навчальних можливостей вивчення вебпрограмування за рахунок розробки і використання спеціалізованої вебсистеми, яка допомагає у вивченні основ вебпрограмування. Проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного програмного застосунку.

Обґрунтовано вибір мови програмування, середовища розробки та технологій, що застосовувалися під час створення навчальної системи. Було розроблено програмний продукт, призначений для вивчення основ вебпрограмування. Для розробки фронтенд-частини застосунку було використано засоби мови програмування JavaScript та фреймворку Vue. Для відображення застосунку використовувалась мова розмітки гіпертексту HTML5. Стилізація інтерфейсу виконувалася за допомогою препроцесора SCSS з використанням фреймворку Bootstrap. Як середовище розробки було використано WebStorm, яке забезпечує зручний та ефективний процес розробки.

Отримані в бакалаврській дипломній роботі результати можна використати для розробки та впровадження навчальної системи, яка буде не лише ефективним засобом вивчення основ вебпрограмування, але й стане інноваційним рішенням у сфері освітніх технологій, що дозволить відкрити нові можливості для студентів та викладачів, які шукають ефективні способи навчання та передачі знань.

Ключові слова: навчальна системи, вебпрограмування, аналіз.

Annotation

The bachelor's thesis consists of 58 pages of A4 format, which includes 34 figures, 3 tables, and a list of references containing 21 items.

The bachelor's thesis analyzes the state of educational systems for learning the basics of web programming. The object, subject, tasks and methods of the study are described. The aim of the research is to improve the learning opportunities for studying web programming through the development and use of a specialized web system that helps in learning the basics of web programming. An analysis of analogues is carried out, their advantages and disadvantages are determined, and the expediency of developing an own software application is proved.

The choice of programming language, development environment, and technologies used to create the educational system is substantiated. A software product has been developed to teach the basics of web programming. To develop the front-end part of the application, the tools of the JavaScript programming language and the Vue framework were used. The HTML5 hypertext markup language was used to display the application. The interface was styled with the SCSS preprocessor using the Bootstrap framework. WebStorm was used as a development environment, which provides a convenient and efficient development process.

The results obtained in the bachelor's thesis can be used to develop and implement a learning system that will not only be an effective tool for learning the basics of web programming, but will also become an innovative solution in the field of educational technology, which will open up new opportunities for students and teachers looking for effective ways to learn and transfer knowledge.

Keywords: educational system, web programming, analysis.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Навчання основ вебпрограмування через різноманітні навчальні системи визначається своєю важливістю у контексті розвитку інформаційних технологій та побудови Інтернет-екосистеми. Знання вебпрограмування стає ключовим елементом в індустрії, що визначає не лише вигляд та функціональність вебсайтів, але й рівень доступу до інформації та інтерактивних сервісів. Навчальні системи призначені для того, щоб студенти отримали необхідні фундаментальні навички у роботі з мовами програмування, вивчили основи вебтехнологій та набули здатності працювати з базами даних [1].

Важливим аспектом таких систем є їхня інтерактивність та практичний підхід. Платформи, які пропонують онлайн-курси, надають можливість студентам використовувати знання, писати код та перевіряти його працездатність в реальному часі.

Актуальність таких навчальних систем обумовлена постійними змінами у технологіях веброзробки. Вони дозволяють студентам та професіоналам відстежувати останні тренди та використовувати найновіші інструменти та технології. Зростання вимог ринку до веброзробників підкреслює актуальність отриманих навичок і підтримує постійний розвиток у цій галузі.

Додатково, вибір теми дослідження зумовлений не лише технологічними та професійними аспектами, але й соціально-економічними факторами. Сучасний ринок праці все більше потребує кваліфікованих веброзробників, і попит на ці навички зростає як у великих IT-компаніях, так і в малих та середніх підприємствах, які прагнуть покращити свою онлайн-присутність. Навчальні системи, що пропонують курси вебпрограмування, допомагають заповнити цю прогалину, надаючи можливість широкому колу людей освоїти необхідні навички для працевлаштування в IT-сфері.

Крім того, важливість теми дослідження підкреслюється необхідністю впровадження інноваційних методик навчання. Традиційні методи освіти часто не

відповідають вимогам сучасного світу, де зміни відбуваються швидкими темпами. Використання інтерактивних навчальних систем дозволяє забезпечити гнучкість навчального процесу, індивідуальний підхід до кожного студента, а також можливість навчатися у зручний для них час та в комфортному середовищі.

Іншою важливою складовою є доступність таких систем для різних верств населення. Онлайн-навчання дає можливість отримувати знання людям з різних регіонів, незалежно від їхнього місця проживання та фінансового становища. Це сприяє зменшенню цифрового розриву та забезпечує рівні можливості для всіх бажаючих навчатися.

З огляду на вищезазначене, дослідження навчальних систем для вивчення основ вебпрограмування є надзвичайно актуальним та важливим для розвитку сучасного суспільства, що активно інтегрується в цифрову еру. Аналіз та покращення таких систем сприятиме підготовці висококваліфікованих фахівців, що в свою чергу підвищить конкурентоспроможність національної економіки на глобальному ринку IT-технологій.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення навчальних можливостей вивчення вебпрограмування за рахунок розробки і використання спеціалізованої вебсистеми, яка допомагає у вивченні основ вебпрограмування.

Задачі дослідження:

- провести аналіз існуючих навчальних систем для вивчення вебпрограмування, визначити їхні переваги та недоліки;
- розробити метод, модель та алгоритми навчальної системи, що враховуватиме потреби та вимоги користувачів, а також сучасні тенденції у веброзробці;
- реалізовувати базовий функціонал вебсистеми, включаючи:
    1) реєстрацію/ аутентифікацію користувачів
    2) перегляд навчальний матеріалів;

3) тестування за темами курсів та інтерактивну область для написання коду;
4) розміщування нових курсів та розробка їх наповнення;

- провести тестування вебсистеми на предмет відповідності встановленим вимогам.

Об’єктом дослідження є процес розробки навчальної системи для вивчення основ вебпрограмування.

Предмет дослідження – методи та засоби реалізації навчальної системи для вивчення основ вебпрограмування.

Новизна отриманих результатів:

1. Подальшого розвитку набув метод додавання та редагування курсів, який, на відміну від існуючих, передбачає можливість оновлення навчальних матеріалів безпосердньо зі сторінки курсу з використанням вебтехнологій Vue.js та TinyMCE (Tiny Moxiecode Content Editor), які дозволяють реалізувати платформо-незалежний створений на JavaScript/HTML WYSIWYG редактор, та використовує систему ролей і прав доступу RBAC (Role-Based Access Control) до створення і редагування навчальних курсів, що покращує наповнення курсів і захист даних навчальної вебсистеми.

2. Подальшого розвитку набула модель роботи навчальної вебсистеми, яка, на відміну від існуючих, орієнтована на використання хмарних технологій AWS, надаючи безсерверні обчислювальні ресурси для реалізації сервера вебсистеми, файлове сховище для зберігання даних користувачів і навчальних курсів, що робить можливим розміщення вебсистеми у мережі Інтернет без надлишкових витрат у випадку відсутності трафіку та дозволяє автоматичне масштабування ресурсів на піках відвідуванності застосунку користувачами, що покращує функціонування системи та робить її економічно ефективною.

Практичне значення полягає в можливості використання розробленої навчальної системи для вивчення основ вебпрограмування і набуття необхідних навичок з розробки вебзастосунків. Ця система допоможе засвоїти основні концепції вебпрограмування, зокрема, роботу з HTML, CSS та JavaScript.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці, опублікованій у співавторстві [2], автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, алгоритм роботи програми.

Апробація результатів роботи. Описані у бакалаврській дипломній роботі положення доповідались на конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)».

Публікації. Результати роботи були опубліковані в матеріалах конференції – LIII Науково-технічної конференції підрозділів Вінницького національного технічного університету (2024) [2].

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 21 літературне джерело.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБСИСТЕМ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБПРОГРАМУВАННЯ

1.1 Аналіз предметної області

Напрямок навчальних систем для вивчення основ вебпрограмування набуває все більшої популярності в сучасному освітньому просторі. Ці системи відрізняються чіткою практичною орієнтацією, забезпечуючи студентам можливість застосовувати теоретичні знання на практиці.

Одна з ключових особливостей таких навчальних систем – використання найсучасніших вебтехнологій, зокрема HTML, CSS, JavaScript, різноманітних фреймворків та бібліотек. Це дозволяє студентам опановувати затребувані на ринку праці навички, які можна одразу застосовувати у реальних проєктах.

Навчальні системи для вивчення вебпрограмування часто ґрунтуються на проектному підході, коли студенти працюють над розробкою вебзастосунків, наближених до реальних. Така практична спрямованість забезпечує комплексне застосування набутих знань та вмінь, формуючи у студентів цілісне розуміння процесу розробки програмного забезпечення.

Важливою характеристикою цих навчальних систем є їх інтерактивність та зворотній зв'язок. Студенти можуть одразу перевіряти результати своєї роботи, отримуючи коментарі та рекомендації від викладачів та однокурсників. Крім того, сучасні онлайн-платформи роблять навчання доступним та гнучким, що дозволяє студентам обирати власну освітню траєкторію.

Актуальність та важливість навчальних систем для вивчення основ вебпрограмування важко переоцінити в сучасних умовах.

По-перше, вебтехнології та програмування є одними з найбільш затребуваних навичок на ринку праці. Практично у будь-якій галузі потрібні фахівці, здатні розробляти вебзастосунки, сайти та інтерактивні онлайн-сервіси. Навчальні системи, що зосереджуються на цій тематиці, допомагають студентам набути необхідних компетенцій та підвищують їх конкурентоспроможність.

По-друге, сучасний світ швидко переходить до цифрового формату, і уміння

розробляти високоякісні продукти стає ключовим елементом загальної цифрової компетенції. Навчальні системи для вивчення основ вебпрограмування сприяють формуванню широкого спектру навичок, необхідних для успішної кар'єри та життя в цифровому суспільстві.

По-третє, такі навчальні системи дозволяють студентам опановувати практичні навички, які вони можуть одразу застосовувати у реальних проєктах. Це суттєво підвищує їхню мотивацію до навчання та забезпечує більш глибоке розуміння предметної області.

Крім цього, навчальні системи для вивчення основ вебпрограмування спроможні адаптуватися до стрімких технологічних змін, пропонуючи актуальний та затребуваний у суспільстві контент. Це дозволяє студентам отримувати найсучасніші знання та навички, які будуть конкурентними на ринку праці.

Отже, актуальною є розробка навчальної системи для вивчення основ вебпрограмування.

1.2 Порівняльний аналіз аналогів

Існує велика кількість систем для вивчення програмування. Розглянемо такі системи, як Codeacademy, Udacity та freeCodeCamp.

Codecademy [3] є онлайн-платформою для навчання програмуванню та інших технічних навичок. Основними особливостями цієї платформи є інтерактивність, доступність та широкий спектр курсів. На відміну від багатьох інших ресурсів, Codecademy пропонує користувачам можливість вивчення програмування через безпосереднє практичне виконання завдань у власному браузері.

Інтерактивні курси Codecademy включають в себе різноманітні мови програмування, такі як Python, JavaScript, Ruby, а також технології веброзробки, такі як HTML та CSS. Кожен урок складається з теоретичної частини, прикладів коду та завдань для самостійного розв'язання. Це сприяє активному залученню студентів та швидкому закріпленню знань.

Codecademy підходить для просунутих користувачів, оскільки він дозволяє швидко вивчати основи програмування та отримувати практичні навички у вигляді

реальних завдань. Однак для глибшого розуміння і розвитку більш складних проектів, може бути корисно доповнювати навчання іншими ресурсами та проектами.

Платформа Codeacademy зображена на рисунку 1.1.

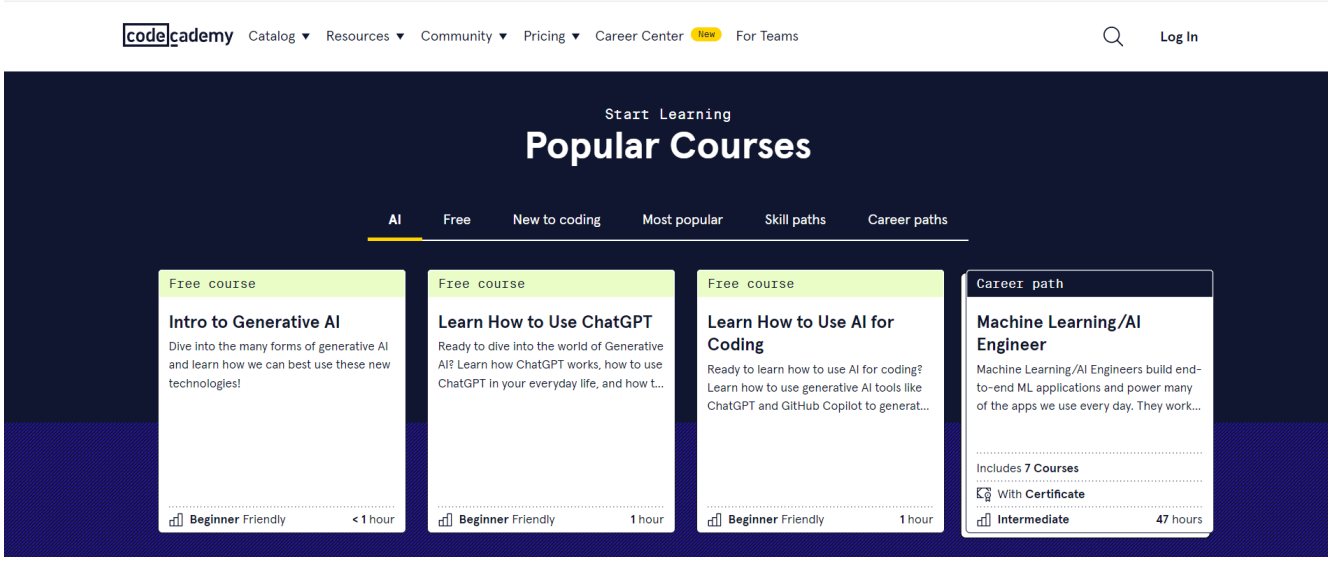


Рисунок 1.1 – Codeacademy

Udacity [4] є онлайн-платформою для навчання з фокусом на технічних навичках та розвитку кар'єри в галузі інформаційних технологій. Однією з ключових особливостей Udacity є акцент на професійному розвитку та високоякісному вмісті, створеному у співпраці з провідними компаніями та експертами галузі. Відзначається відмінною якістю викладання та можливістю отримання відмітних сертифікатів та дипломів.

Udacity пропонує курси з різних областей, включаючи програмування, штучний інтелект, веброзробку, аналіз даних, машинне навчання та інші. Кожен курс побудований як повноцінний навчальний досвід, з відеоуроками, практичними завданнями та проектами, що дозволяють студентам застосовувати отримані знання у реальних сценаріях.

Однією з важливих особливостей Udacity є акцент на практичному застосуванні вивченого матеріалу через проекти, які зазвичай розробляються у співпраці з технологічними компаніями-партнерами. Це надає студентам

можливість побудувати портфоліо реальних проектів та отримати практичний досвід, що сприяє підвищенню їхніх шансів на знаходження роботи.

Udacity також пропонує програми "Nanodegree", які є поглибленими навчальними курсами, розробленими для конкретних областей, і завершуються видачею офіційного сертифіката або диплому. Ці програми часто мають тривалість та вартість, але надають більш глибокий рівень експертизи та можуть бути особливо корисними для тих, хто прагне розвивати конкретні навички в глибоких технічних областях.

Платформа Udacity продемонстрована на рисунку 1.2.

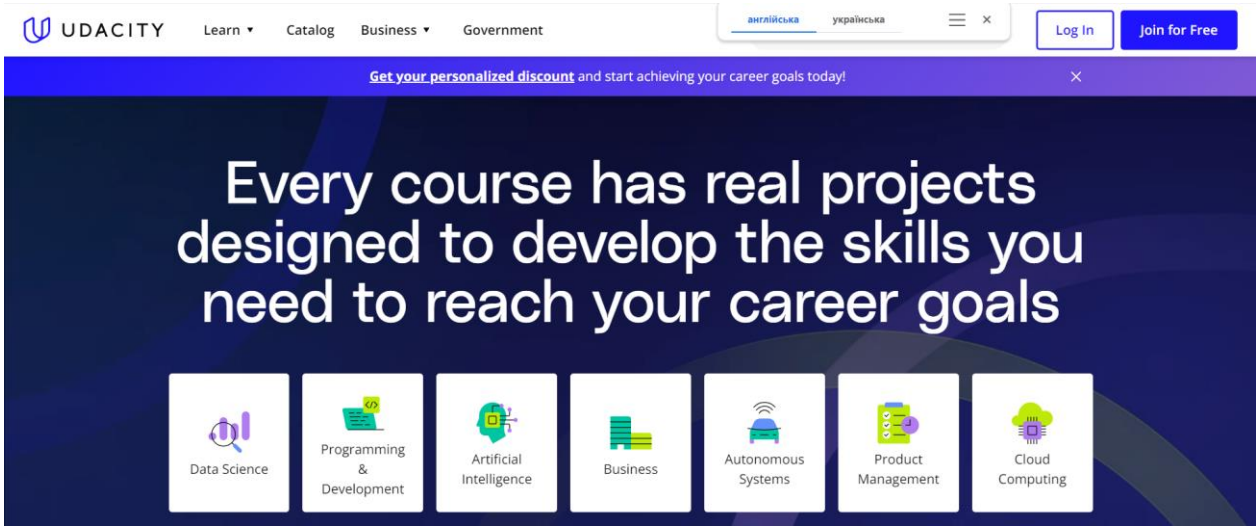


Рисунок 1.2 – Udacity

freeCodeCamp [5] – це безкоштовна онлайн-навчальна платформа, призначена для того, щоб допомагати людям навчатися програмуванню та розвивати навички веброзробки. Заснована на ідеї відкритого доступу та взаємодопомоги в спільноті, freeCodeCamp визначається своєю місією допомогти кожній людині отримати доступ до якісної технічної освіти.

Основним елементом freeCodeCamp є її програма Certification, яка включає широкий спектр курсів з веброзробки та програмування. Навчання на платформі базується на практичних завданнях, які допомагають студентам застосовувати здобуті знання.

Однією з переваг freeCodeCamp є його велика та активна спільнота

користувачів. Студенти можуть обговорювати питання, ділитися досвідом та отримувати підтримку в чаті та форумах. Важливою частиною платформи є можливість виконання практичних проектів для неприбуткових організацій, що дозволяє студентам набувати досвід роботи над реальними завданнями та внести свій внесок у суспільство.

Платформа freeCodeCamp продемонстрована на рисунку 1.3.

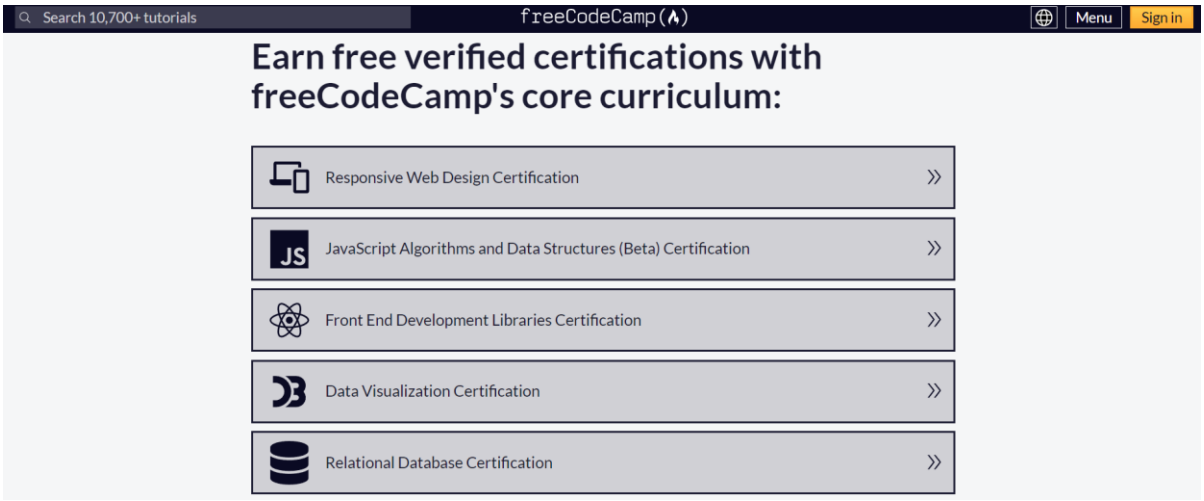


Рисунок 1.3 – FreeCodeCamp

Було створено таблицю порівняння, в яких співставлені характеристики описаних систем та власної розробки (таблиця 1.1).

Таблиця 1.1 – Порівняння характеристик аналогів

Критерій	Codeacademy	Udacity	freeCodeCamp	Власна розробка
1	2	3	4	5
Безкоштовний доступ	0	0	1	1
Спільнота користувачів	1	0	1	1
Проходження практичних завдань користувачами	0	1	0	1
Проходження питань для самоконтролю користувачами	1	1	1	1

Продовження таблиці 1.1.

1	2	3	4	5
Відслідковування навчального прогресу та результатів тестування	1	1	1	1
Сумарний коефіцієнт	3	3	4	5

Таким чином, власна розробка має вищий сумарний коефіцієнт за Codeacademy, Udacity на 40% (100% – 3/5*100% = 40%), за freeCodeCamp на 20% (100% – 4/5*100% = 20%).

1.3 Аналіз методів розв’язання задачі

Однією з ключових концепцій при розробці навчальної системи для вивчення основ вебпрограмування є модульна архітектура. Тут застосовується об'єктно-орієнтоване програмування та принципи модульності, щоб створити гнучку та масштабовану систему. Кожен модуль відповідає за певний аспект вебпрограмування, таким чином розділяючи логіку презентації, бізнес-логіку та взаємодію з даними.

Окрім модульного підходу, існують й інші. Одним з таких є об'єктно-орієнтований. Застосування принципів ООП, таких як інкапсуляція, успадкування та поліморфізм, дозволить створити структуровану та гнучку систему. Об'єктно-орієнтоване проектування може спростити розширення функціоналу та підтримку системи в майбутньому.

Компонентний підхід передбачає розробку системи з використанням повторно використовуваних компонентів, кожен з яких відповідає за певну функціональність. Це дасть можливість швидко розробляти та впроваджувати нові можливості.

Ще одним варіантом є сервісно-орієнтована архітектура (SOA) [6], яка організовує систему у вигляді набору сервісів, що взаємодіють між собою за допомогою чітко визначених інтерфейсів. Така архітектура дозволить досягти

високої гнучкості, масштабованості та можливості повторного використання компонентів.

Мікросервісна архітектура передбачає розділення системи на невеликі, незалежні сервіси, кожен з яких відповідає за певну бізнес-функцію. Така структура полегшить розгортання, масштабування та незалежне оновлення окремих компонентів системи.

Серед цих підходів для реалізації навчальної системи для вивчення вебпрограмування, найбільше виділяється модульний підхід.

Модульна архітектура дозволить розділити систему на окремі, взаємопов'язані модулі, які відповідають за різні аспекти вебпрограмування (HTML, CSS, JavaScript, серверна частина тощо). Це забезпечить гнучкість, масштабованість та можливість незалежного розвитку кожного модуля.

Модульність сприяє застосуванню принципів об'єктно-орієнтованого програмування, що дає можливість створити стійку, модифіковану та розширювану систему.

Розділення інтерфейсу, бізнес-логіки та взаємодії з даними, притаманне модульному підходу, добре відповідає вимогам до такої навчальної системи.

Модульна структура дозволить легко інтегрувати в систему сучасні фреймворки та бібліотеки веброзробки (React, Angular, Node.js тощо), що сприятиме використанню передових технологій та забезпечить високу якість розробки.

Модульність полегшить впровадження адаптивного та персоналізованого навчання, оскільки кожен модуль можна налаштовувати окремо відповідно до потреб користувачів.

Таким чином, для розробки власної навчальної системи для вивчення основ вебпрограмування найкращою є модульна архітектура.

1.4 Постановка задач дослідження

У результаті аналізу предметної області навчальних систем для вивчення основ вебпрограмування, порівняння існуючих аналогів та вивчення їх

можливостей, аналізу методів розв’язання задачі, було вирішено розробити власну навчальну систему для вивчення основ вебпрограмування та визначено завдання, які необхідно виконати для розробки навчальної системи:

- провести аналіз існуючих навчальних систем для вивчення вебпрограмування, визначити їхні переваги та недоліки;
- розробити метод, модель та алгоритми навчальної системи, що враховуватиме потреби та вимоги користувачів, а також сучасні тенденції у веброзробці;
- реалізовувати базовий функціонал вебсистеми, включаючи:
    5) реєстрацію/ аутентифікацію користувачів
    6) перегляд навчальний матеріалів;
    7) тестування за темами курсів та інтерактивну область для написання коду;
    8) розміщування нових курсів та розробка їх наповнення;
- провести тестування вебсистеми на предмет відповідності встановленим вимогам.

Робота передбачає створення декількох ключових модулів, які будуть інтегровані в єдину вебсистему. Ці модулі включають:

- модуль управління даними, який відповідає за зберігання, обробку та надання доступу до інформації, необхідної для навчального процесу.
- модуль графічного інтерфейсу, який забезпечує зручний та інтуїтивно зрозумілий спосіб взаємодії користувачів з системою.

Згідно з метою роботи, необхідно розробити такий функціонал вебсистеми:

1. Систему автентифікації та авторизації для користувачів.
2. Механізм розміщення навчальних матеріалів, здійснюваного адміністраторами.
3. Функціонал для розміщення відеоматеріалів, який також керується адміністраторами.
4. Інструмент для розміщення питань самоконтролю, створюваних адміністраторами.

5. Можливість проходження питань самоконтролю користувачами.

6. Функцію відслідковування навчального процесу та результатів тестування для користувачів.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було проведено аналіз предметної області програмних систем для вивчення основ вебпрограмування. Було встановлено, що напрямок вивчення вебпрограмування є популярним та таким, що стрімко розвивається. В результаті цього було зроблено висновок, що розробка власної системи є актуальною. Було проведено порівняльний аналіз аналогів, такий як FreeCodeCamp, Udacity, Codeacademy, які уже існують на ринку навчальних систем, співставлено їхні можливості та можливості власної системи, що розробляється, в результаті чого було доведено конкурентоздатність та актуальність розробки. Було проведено аналіз методів розв’язання задачі та прийнято рішення розробити систему за модульною архітектурою. В результаті, було проведено постановку задач дослідження та сформульовано можливості, які надаватиме користувачам розроблювана система.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ

2.1 Аналіз даних

Дані та їх обробка – це те, без чого сьогодні не можна уявити жодну програмну систему. Великі потоки даних генеруються самими системами залежно від їхніх можливостей, призначення та потреб користувача. Окрім цього, самі користувачі вводять значні обсяги даних для взаємодії з програмними системами для задоволення своїх потреб.

Розроблювана навчальна система для вивчення основ вебпрограмування не є виключенням. Так, при реєстрації в системі зберігаються та обробляються такі дані про користувача, як: ім’я користувача, адреса електронної пошти та пароль користувача. Ім’я користувача необхідне для ідентифікації його в системі та для створення окремого запису в базі даних. Електронна адреса використовується на випадок необхідності в надсиланні електронних листів та для підтвердження автентичності створюваного облікового запису. Пароль використовується для контролю доступу до облікового запису.

Так як користувач проходить певні курси, що займає не один його візит до системи та не один день, важливо зберігати його прогрес, місце, де він зупинився під час останнього входу в систему. Інформація про курси, що проходить користувач та прогрес проходження зберігається в базі даних системи та прикріплюється до аккаунту користувача.

Після проходження курсів, користувач проходить тестування для перевірки отриманих знань. Для цього йому надається відповідний опитувальник, який він проходить. Так як система заздалегідь знає правильні відповіді, вона співставляє їх з тими відповідями, що надав користувач, та розраховує результат проходження тесту.

Окрім власне даних, що стосується користувача, також обробляються та зберігаються дані про курси, їх зміст, призначення, тривалість та набір навчальних матеріалів/відео, які можуть переглядати та проходити користувачі.

2.2 Розробка моделі системи

Розробка моделі системи несе в собі кілька ключових цілей. Вона надає структуроване подання про те, як різні сервіси AWS взаємодіють один з одним у межах інфраструктури. Вона документує загальну архітектуру системи, робить її візуально зрозумілою для всіх зацікавлених сторін.

Модель системи відображає, як дані рухаються між різними компонентами, такими як Amazon Cognito, Active Directory, Amazon DocumentDB тощо. Це дозволяє краще зрозуміти логічні залежності та потоки інформації. Маючи таке наочне уявлення про архітектуру, розробники, архітектори та менеджери можуть ефективніше спілкуватися та узгоджувати рішення. Діаграма слугує загальною "мовою", якою вони можуть описувати та обговорювати систему.

Модель системи допомагає розуміти, які компоненти можуть бути розширені, модернізовані чи замінені в міру зростання та розвитку системи. Вона дає загальну картину, яка підтримує процес прийняття рішень щодо масштабування та вдосконалення архітектури.

Модель системи для вивчення основ вебпрограмування наведено на рис. 2.1.

Її складові компоненти:

1. Users – користувачі системи.
2. Amazon Cognito [7] – служба, яка відповідає за авторизацію та автентифікацію користувачів.
3. Active Directory – сервіс, який надає можливість керувати ідентифікацією та доступом користувачів.
4. Amazon DocumentDB [8] – база даних, що використовується для зберігання документів. Вона являє собою дистрибутив СКБД MongoDB від Amazon.
5. Amazon API Gateway – шлюз для доступу до API.
6. Lambda – serverless-обчислення, які виконують певні функції.
7. CloudWatch – служба моніторингу та управління ресурсами AWS.
8. Amazon S3 [9] – сховище об'єктів для зберігання даних.

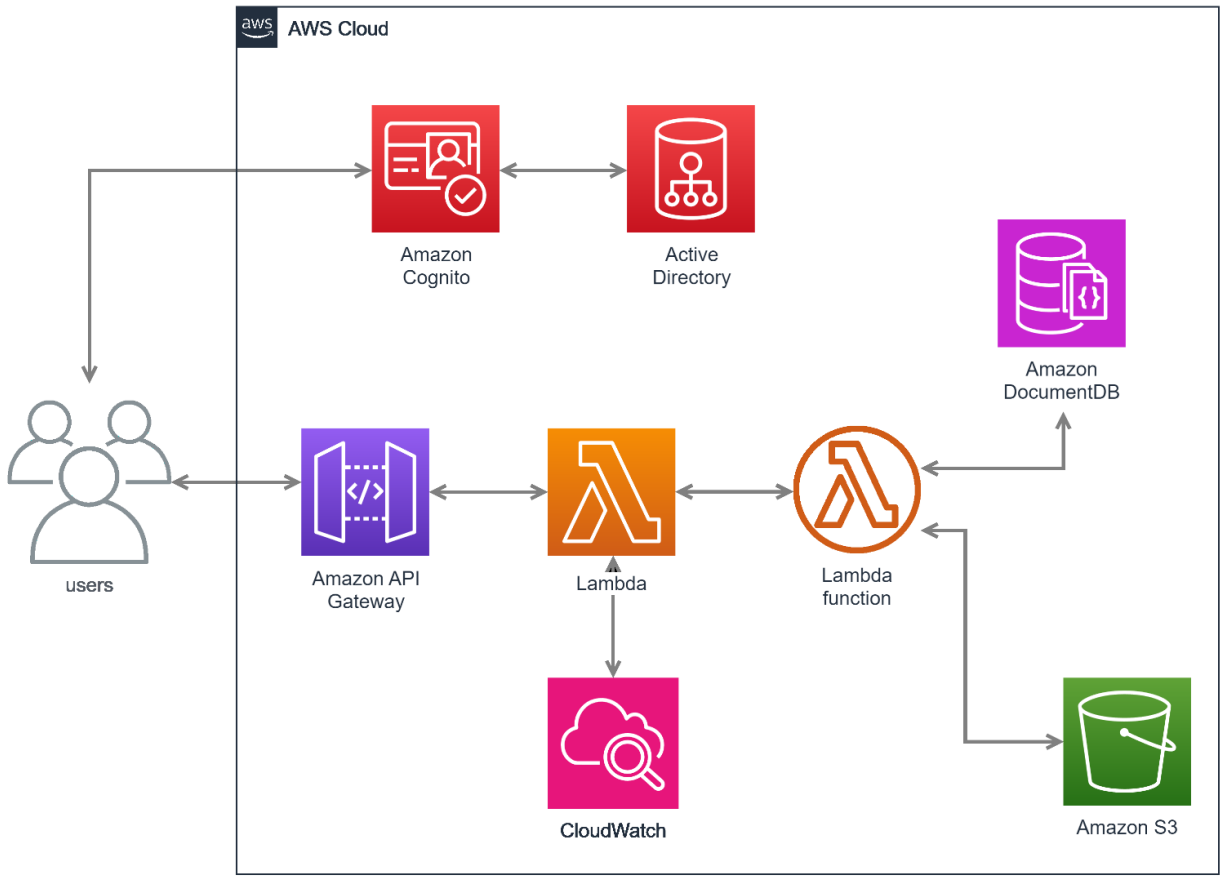


Рисунок 2.1 – Модель системи

2.3 Розробка методу додавання та редагування курсів з авторизацією

Додавання та редагування курсів адміністраторами системи є обов'язковою властивістю вебзастосунку навчальної системи орієнтованої на вивчення основ вебпрограмування. Також не менш важливою є авторизація, яка запобігає несанкціонованому редагуванню, чи додаванню курсів без відома адміністратора платформи. Авторизація здійснюється через надання користувачам відповідних ролей, перевірки їх відповідності та належність курса до авторизованої особи. Таким чином система перешкоджатиме створенню курсів звичаним користувачам, надаючи цю можливість виключно викладачам, лекторам, авторам курсів та водночас запобігатиме редагуванню курсів особами, що не є їх безпосередніми авторами.

Метод додавання та редагування курсів з авторизацією складається з таких кроків:

1. Користувач використовує метод authenticate класу User для перевірки своїх облікових даних (ім'я користувача та пароль), збережених у базі даних (users_db). Якщо дані вірні, користувач отримує доступ до системи.

2. Перевірка прав доступу користувача відбувається автоматично під час аутентифікації. Якщо користувач має роль 'admin', він має право створювати нові курси.

3. Користувач натискає на кнопку "Додати новий курс", спрацьовує метод add_course у головній програмі, який переадресовує користувача на сторінку створення курсу.

4. Користувач заповнює поля курсу: назву (title), опис (description) та кількість годин (hours). Ця інформація зберігається у локальних змінних об'єкта course.

5. Для кожного розділу викликається метод add_section класу Course, який додає новий розділ з назвою та списком параграфів. Параграфи додаються за допомогою методу add_paragraph, який додає параграф з назвою та описом до поточного розділу.

6. Для кожного параграфа викликається метод add_question, який додає тестове питання з варіантами відповідей. Правильний варіант відмічується спеціальним прапорцем.

7. Зберігання курсу: Після завершення всіх кроків, користувач натискає на кнопку "Додати курс". Головна програма викликає метод save_courses, який зберігає інформацію про курс у форматі JSON у файлі courses.json через виклик HTTP методу POST /courses.

Алгоритм роботи методу додавання курсів та редагування курсів із авторизацією наведено на рисунку 2.2.

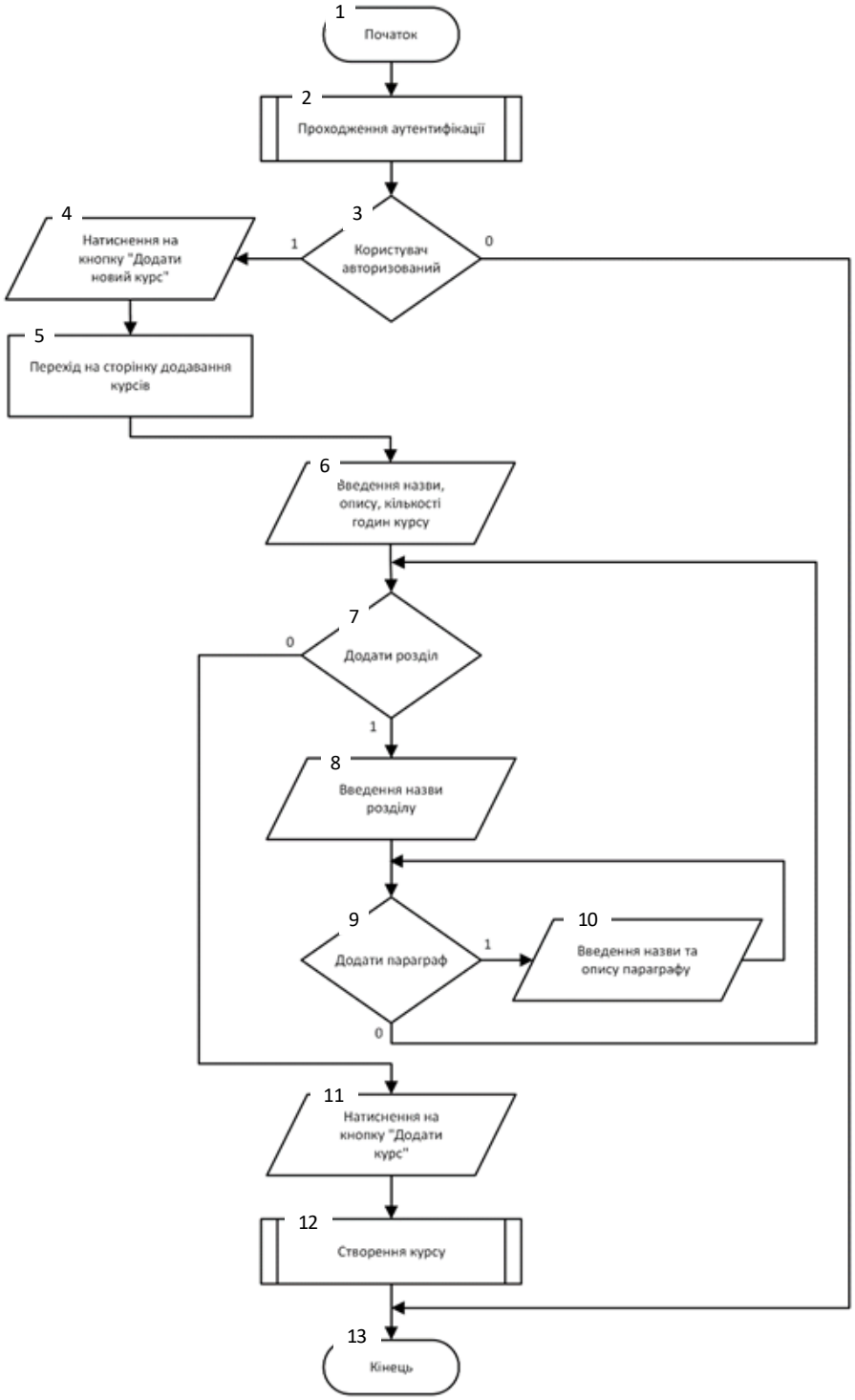


Рисунок 2.2 – Блок-схема алгоритму роботи методу додавання курсів

2.4 Розробка загальних алгоритмів роботи застосунку

Загальний алгоритм роботи додатку описує основні етапи та кроки, які виконує програмний додаток під час своєї роботи. Він визначає загальну структуру та логіку функціонування застосунку на високому рівні.

Ця блок-схема відображає такі дії користувача, як реєстрація або

автентифікація, залежно від того, чи зареєстрований користувач, та перехід між відповідними модулями системи залежно від вибору користувача. Крім того, вона включає обробку запитів та відповідей, взаємодію з базою даних і виконання серверних скриптів, що забезпечує ефективну роботу додатку та задоволення потреб користувачів.

Блок-схема загального алгоритму роботи навчальної системи для вивчення основ вебпрограмування зображено на рисунку 2.3.

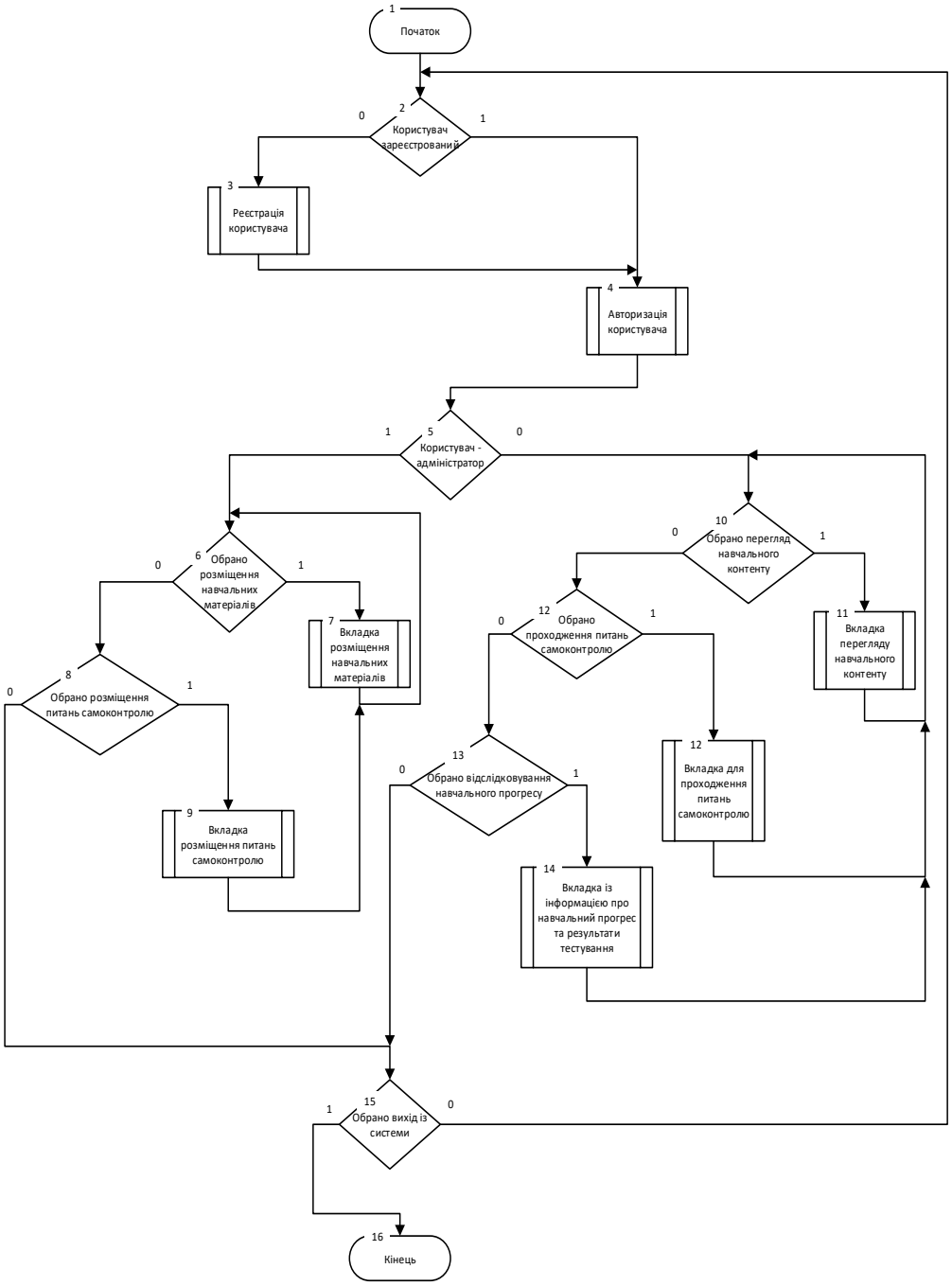


Рисунок 2.3 – Блок-схема загального алгоритму роботи системи

Реєстрація є першим етапом, який проходить користувач перед використанням системи. Він має можливість зареєструватися, використовуючи логін та пароль. Процес реєстрації складається з кількох кроків:

1. Введення персональних даних: користувач заповнює поля форми реєстрації, де вказує своє ім'я, електронну пошту, логін та пароль. Можливо, система потребує підтвердження пароля, щоб уникнути помилок під час введення.

2. Вибір логіну та пароля: користувач вибирає унікальний логін, який буде використовуватися для входу в систему, та створює надійний пароль. Пароль має відповідати певним вимогам безпеки, таким як мінімальна довжина, наявність цифр та спеціальних символів.

Блок-схема процесу реєстрації користувача наведена на рисунку 2.4.

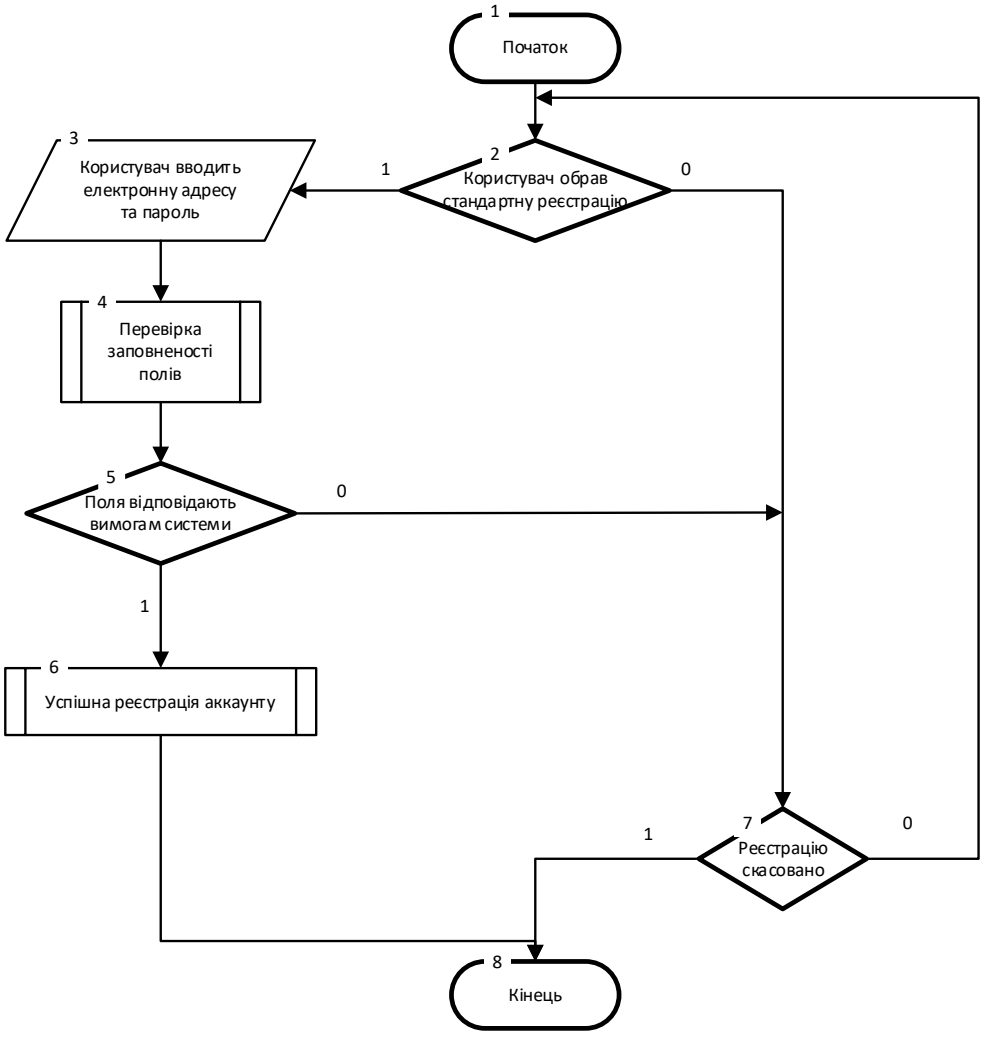


Рисунок 2.4 – Блок-схема процесу реєстрації користувача

Після успішної реєстрації користувача наступним кроком є авторизація. Це процес перевірки автентичності, де користувач надає свої облікові дані (Email та пароль), щоб отримати доступ до системи або ресурсу.

Блок-схема процесу авторизації наведена на рисунку 2.5.

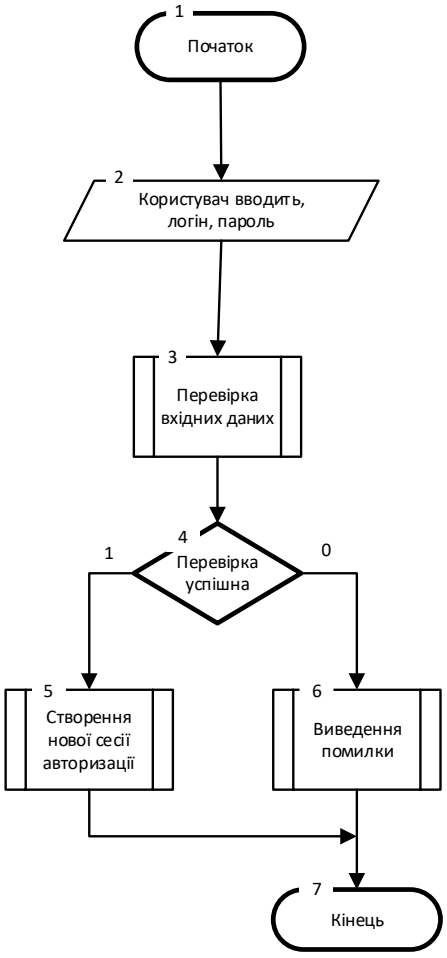


Рисунок 2.5 – Блок-схема процесу авторизації

2.5 Висновки

У другому розділі, було проведено аналіз даних, що використовуються та обробляються системою для її функціонування та задоволення потреб користувача. Розроблено модель системи для розуміння її архітектури та полегшення розробки. Розроблено основні алгоритми роботи системи.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ

3.1 Варіантний аналіз та обґрунтування вибору мови програмування

JavaScript [10] – це динамічна, інтерпретована, об'єктно-орієнтована мова програмування, яка широко використовується для створення інтерактивних вебсторінок і вебзастосунків.

Основною особливістю JavaScript є те, що він може використовуватися як на стороні клієнта (у браузері), так і на стороні сервера (за допомогою платформи Node.js). Це дозволяє створювати повноцінні вебзастосунки, де логіка виконується як на клієнті, так і на сервері.

JavaScript є динамічною мовою, що означає, що змінні можуть приймати різні типи даних під час виконання програми. Це надає гнучкість у програмуванні, але водночас вимагає уважності від розробників, щоб уникнути неочікуваної поведінки коду.

Окрім динамічності, JavaScript також підтримує об'єктно-орієнтовану парадигму програмування. Це дозволяє створювати складні структури даних і логіки, використовуючи концепції класів, об'єктів, успадкування тощо. JavaScript також має функціональні можливості, такі як анонімні функції, замикання та вищі порядкові функції.

Важливою рисою JavaScript є його асинхронність. Завдяки цьому вебсторінки можуть залишатися інтерактивними і не "замерзати" під час виконання довготривалих операцій. Асинхронність реалізується в JavaScript за допомогою таких концепцій, як колбеки, проміси та async/await.

Розглянемо переваги та недоліки JavaScript.

Переваги JavaScript:

1. JavaScript можна використовувати як для клієнтської, так і для серверної розробки. Це робить його дуже гнучким та всебічним інструментом.

2. JavaScript дозволяє створювати динамічні, інтерактивні та responsive вебсторінки, що підвищує залученість користувачів.

3. Існує величезна кількість бібліотек, фреймворків та інструментів, які значно спрощують та прискорюють розробку вебзастосунків на JavaScript.

Недоліки JavaScript:

1. JavaScript, як мова, виконувана на клієнті, може становити потенційні ризики безпеки, такі як міжсайтовий скриптинг (XSS) та ін.

2. У деяких випадках, особливо для обчислювально-інтенсивних завдань, JavaScript може поступатися іншим мовам у продуктивності.

3. Динамічна природа JavaScript може ускладнювати налагодження коду та рефакторинг великих проектів.

Незважаючи на деякі недоліки, JavaScript залишається надзвичайно потужною та затребуваною мовою програмування, особливо для веброзробки.

JavaScript є найкращим вибором серед перерахованих мов для розробки навчальної системи з вивчення основ вебпрограмування з кількох причин.

JavaScript є мовою, яка може використовуватися як на стороні клієнта (у браузері), так і на стороні сервера (за допомогою Node.js). Це дозволить створити повноцінну вебсистему, де логіка виконуватиметься як на клієнтській стороні (для забезпечення інтерактивності та динамічності), так і на серверній (для обробки даних, взаємодії з базами даних тощо). PHP [11] та C# [12] зазвичай використовуються лише для серверної частини.

Одна з ключових переваг JavaScript є можливість використання Node.js [13] для розробки серверної частини. Node.js дозволяє розробляти серверні застосунку та API [14], використовуючи JavaScript, що значно спрощує процес навчання та розробки для студентів, які вже знайомі з JavaScript на стороні клієнта. Це забезпечує повну сумісність та спрощує перехід між клієнтською та серверною частинами вебзастосунку.

3.2 Розробка модуля для авторизації та реєстрації користувача

Модуль автентифікації в розробленій веб-системі для вивчення основ вебпрограмування спроектований таким чином, щоб надати користувачам простий та зручний спосіб реєстрації та входу до системи. Коли користувач відкриває

застосунок, він зустрічає головне вікно, на якому йому надається вибір між реєстрацією нового профілю або входом до системи з використанням своїх унікальних даних (див. рисунок 3.1)

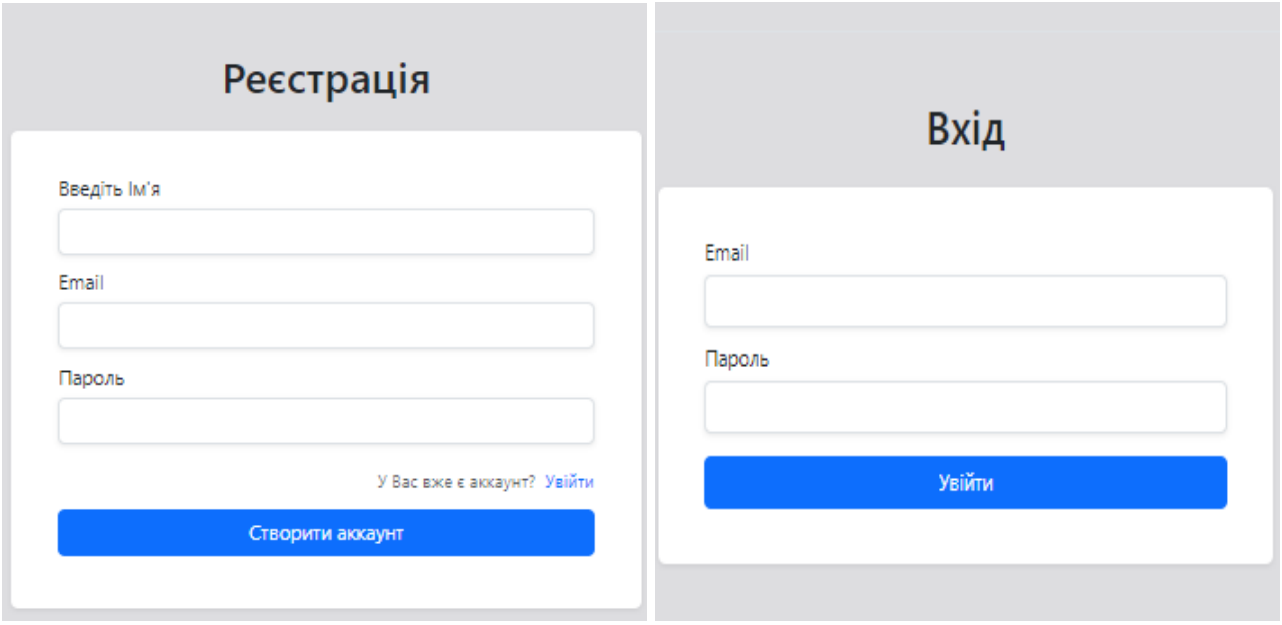


Рисунок 3.1 – Схема інтерфейсу сторінок авторизації та реєстрації

Модуль авторизації дозволяє користувачам увійти в систему, використовуючи свої облікові дані. Він складається з форми входу, де користувач повинен ввести свій електронний адрес та пароль. Після перевірки цих даних сервером, якщо вони валідні, користувач отримує доступ до своїх облікових даних та функцій системи.

Форма входу реалізована за допомогою Vue.js [15], де ‘v-model’ використовується для двостороннього зв'язування даних з полями форми. Це означає, що будь-які зміни, внесені користувачем в поля форми, автоматично оновлюють відповідні змінні в даних компонента. Коли користувач натискає кнопку відправки форми, викликається метод ‘axios.post’ [16], який відправляє дані форми на сервер у вигляді JSON-об'єкта [17]. Запит містить поля email та password, які відповідають значенням полів форми.

На стороні сервера відбувається перевірка введених даних. Якщо вони валідні та співпадають з даними, збереженими в базі даних MongoDB [18], сервер генерує токен авторизації та відправляє його назад клієнту. Токен потім використовується

для встановлення сесії користувача, що дозволяє йому мати доступ до обмежених ресурсів додатку.

Якщо відповідь від сервера позитивна (код статусу 200), користувач переходить на головний екран за допомогою ‘this.$router.push('/')’, що забезпечує плавний перехід до основної частини додатку. У випадку негативної відповіді (наприклад, код статусу 401 [19] для небажаних запитів) відображається повідомлення про помилку, яке інформує користувача про проблему з його обліковими даними (див. рисунок 3.2).

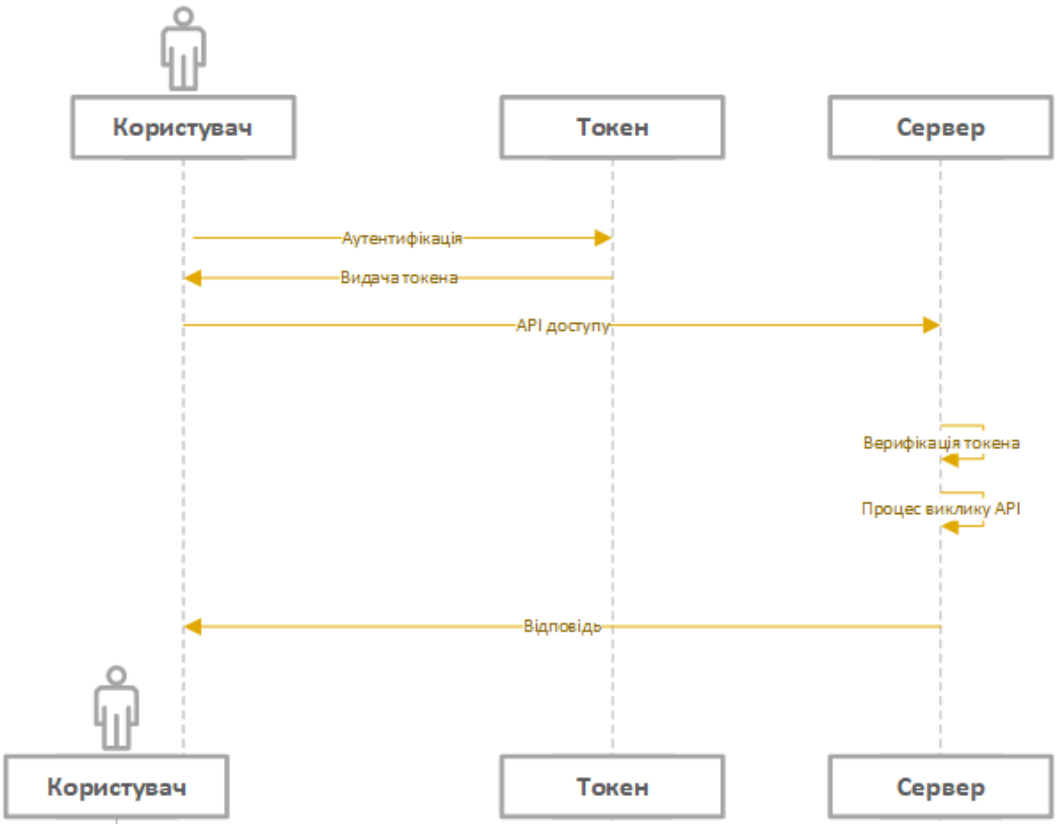


Рисунок 3.2 – Діаграма послідовності авторизації користувача

Модуль реєстрації дозволяє новим користувачам створити обліковий запис у системі. Користувачі мають введення своєї інформації, такої як ім'я, електронна пошта та пароль, яку потім сервер зберігає та використовує для подальшої ідентифікації.

Форма реєстрації також реалізована за допомогою Vue.js, де ‘v-model’ використовується для двостороннього зв'язування даних з полями форми. Це

забезпечує синхронізацію між полями форми та даними компонента, дозволяючи автоматично оновлювати дані компонента при зміні значень в полях форми.

Коли користувач натискає кнопку відправки форми, викликається метод ‘axios.post’, який відправляє дані форми на сервер у вигляді JSON-об'єкта. Запит містить поля name, email та password, які відповідають значенням полів форми.

На стороні сервера обробляється цей запит. Якщо дані валідні, сервер створює новий обліковий запис з введеною інформацією та зберігає його в базі даних. Після створення облікового запису сервер відправляє відповідь клієнту про успішну реєстрацію.

Якщо відповідь від сервера позитивна, користувач переходить на сторінку входу за допомогою ‘this.$router.push('/login')’, що дозволяє йому увійти в систему з новоствореним обліковим записом. У випадку негативної відповіді відображається повідомлення про помилку, яке інформує користувача про проблему з реєстрацією.

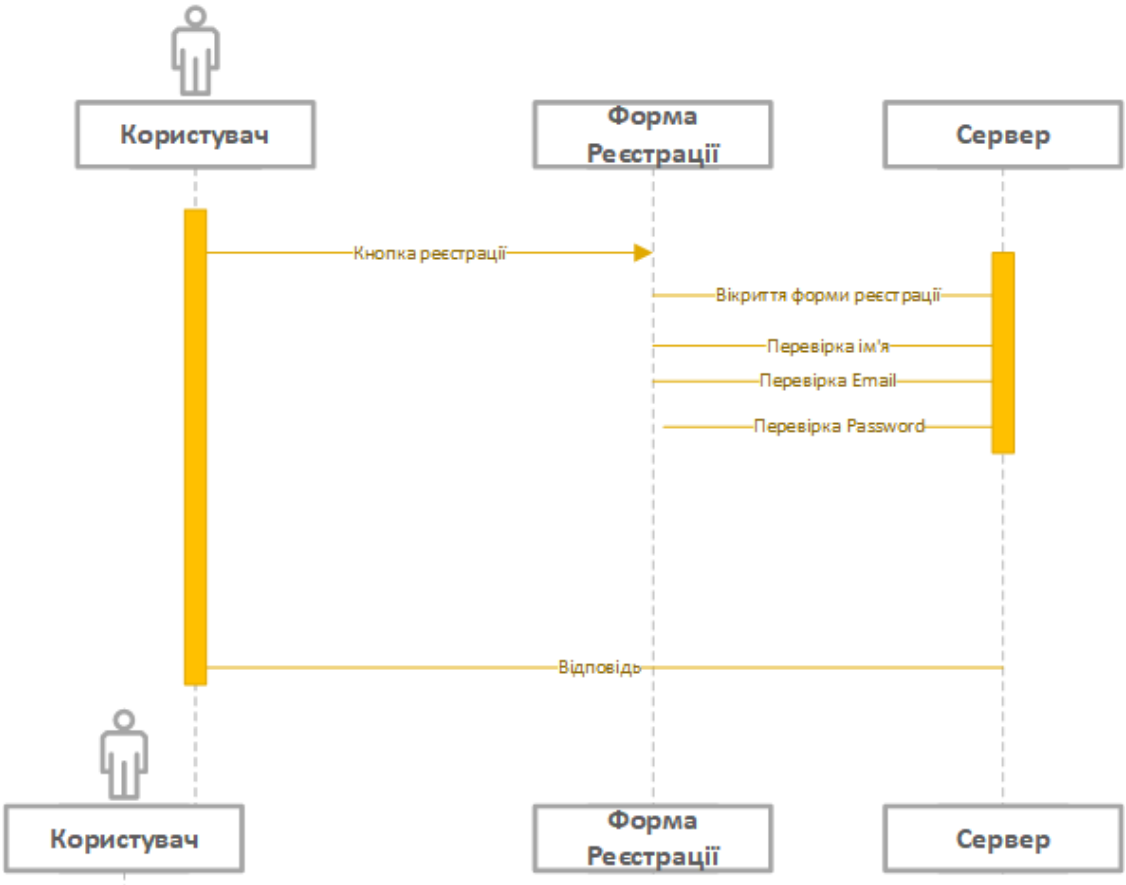


Рисунок 3.3 – Діаграма послідовності реєстрації користувача

Для взаємодії користувачів із базою даних системи було розроблено структуру об’єктів відповідно до їх ролей у системі, як на рисунку 3.4.

Рисунок 3.4 – Структура об’єкта користувача

Опис об’єкта:

- id – ідентифікаційний номер, унікальний для кожного запису;

- name – ім’я користувача;

- email – електронна скринька користувача для входу/реєстрації у системі;

- password – пароль користувача для входу у систему;

- roles – ролі користувачів;

Ці об'єкти дають змогу користувачеві увійти в систему та здійснювати перегляд курсів відповідно до їх потреб.

3.3 Розробка модуля для створення курсів

Важливою складовою навчальних вебсистем є можливість розміщення матеріалів для навчання за що відповідає розроблений модуль створення курсів,

який є доступним лише для авторизованих користувачів.

Наділений відповідними правами доступу користувач системи має можливість додавання нових курсів та редагування вже існуючих, які перебувають у його авторській власності. Для створення курсу необхідно виконати вхід у систему, та натиснути на доступну лише авторизованим користувачам кнопку «Додати новий курс» (див. рисунок 3.5).

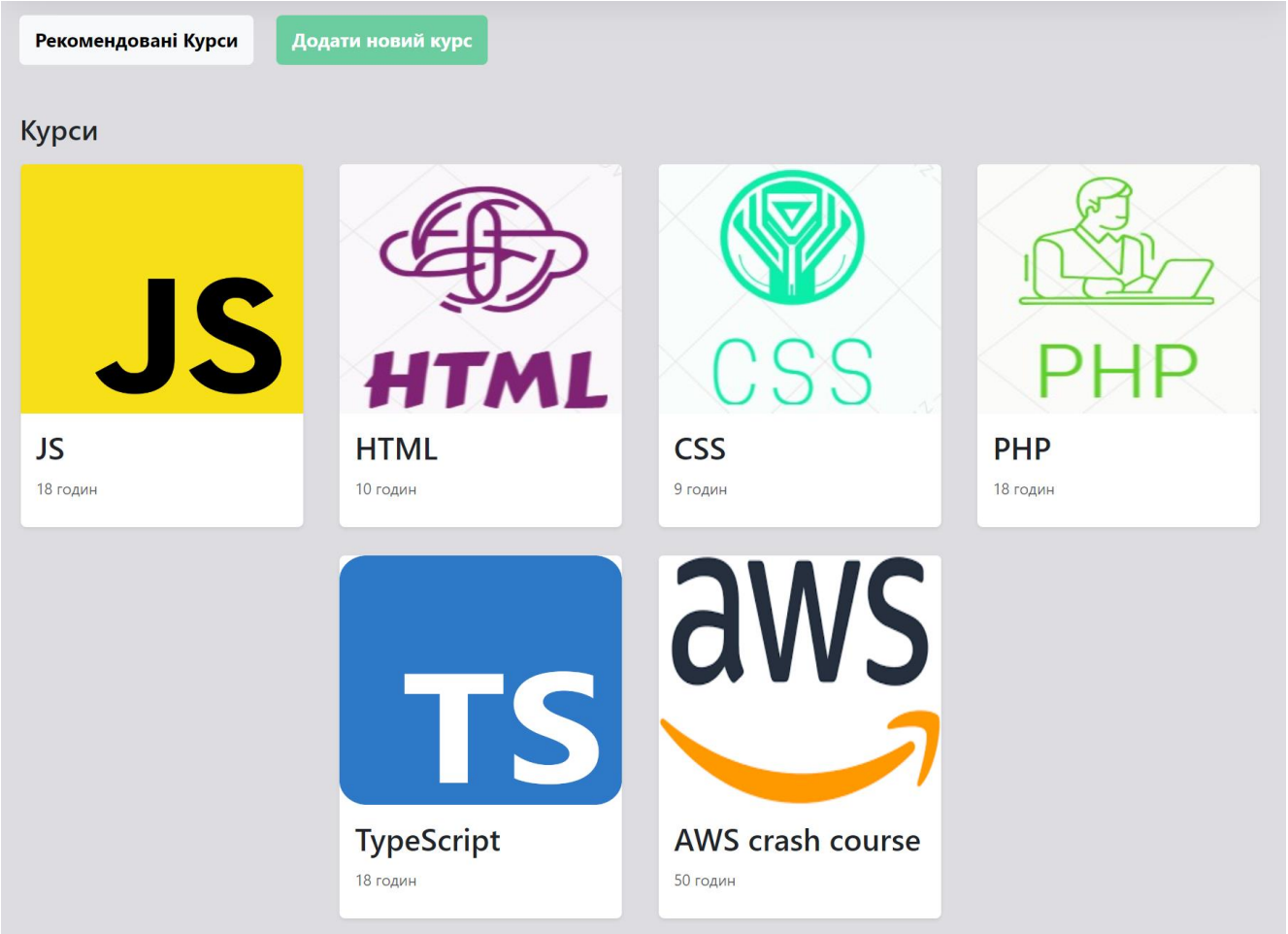


Рисунок 3.5 – Додати новий курс

Після натиснення, користувач буде переведений на сторінку із формою введення, де він може додавати (див. рисунок 3.6) опис курсу, зображення, розділи та відповідні ним параграфи.

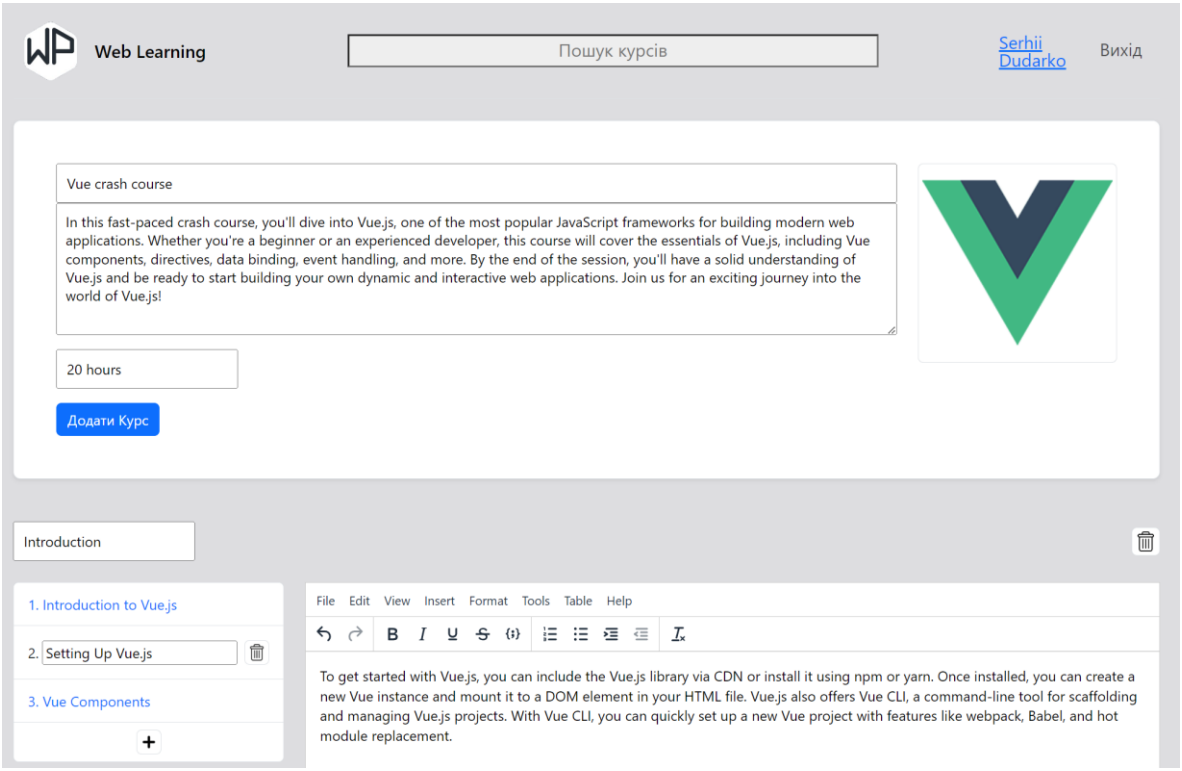


Рисунок 3.6 – Додавання нового курсу

Користувач може зберегти новостворений курс після заповнення всіх даних, натисканням на кнопку «Додати курс», після чого буде виконаний HTTP POST [20] запит на сервер на збереження курсу у форматі JSON (див. рисунок 3.7).

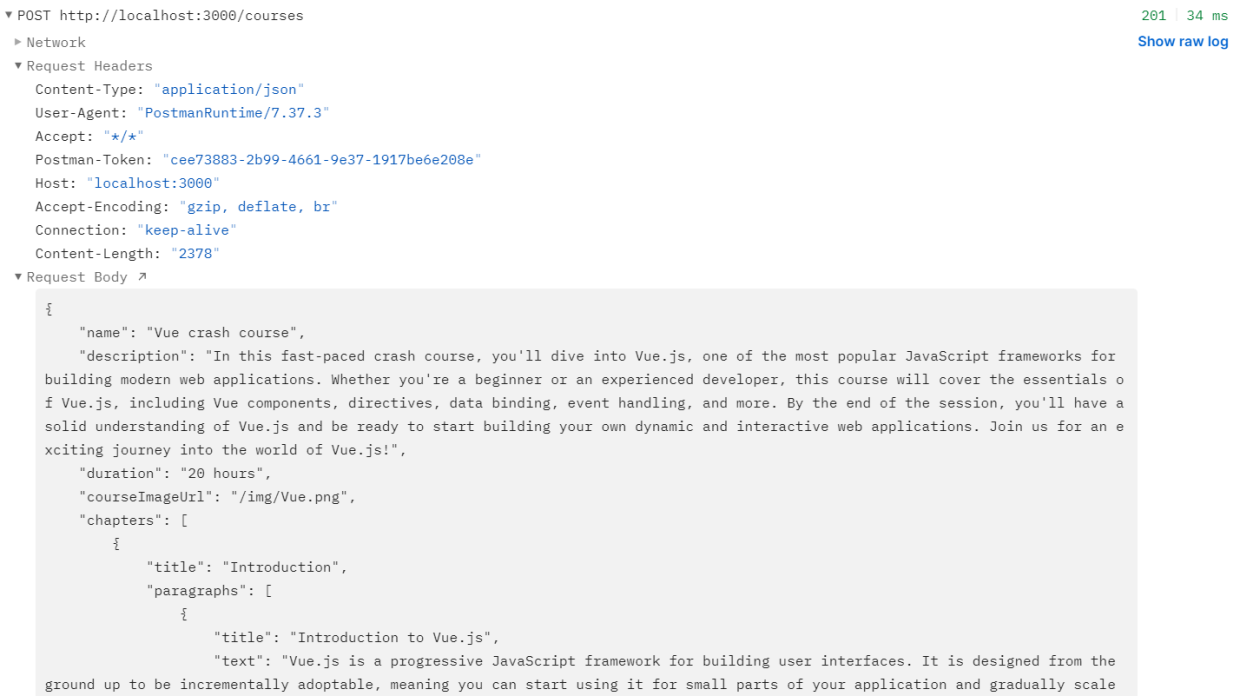


Рисунок 3.7 – Виконання запиту додавання курсу

Результатом запиту є запис у базі формату JSON (див. рисунок 3.8).

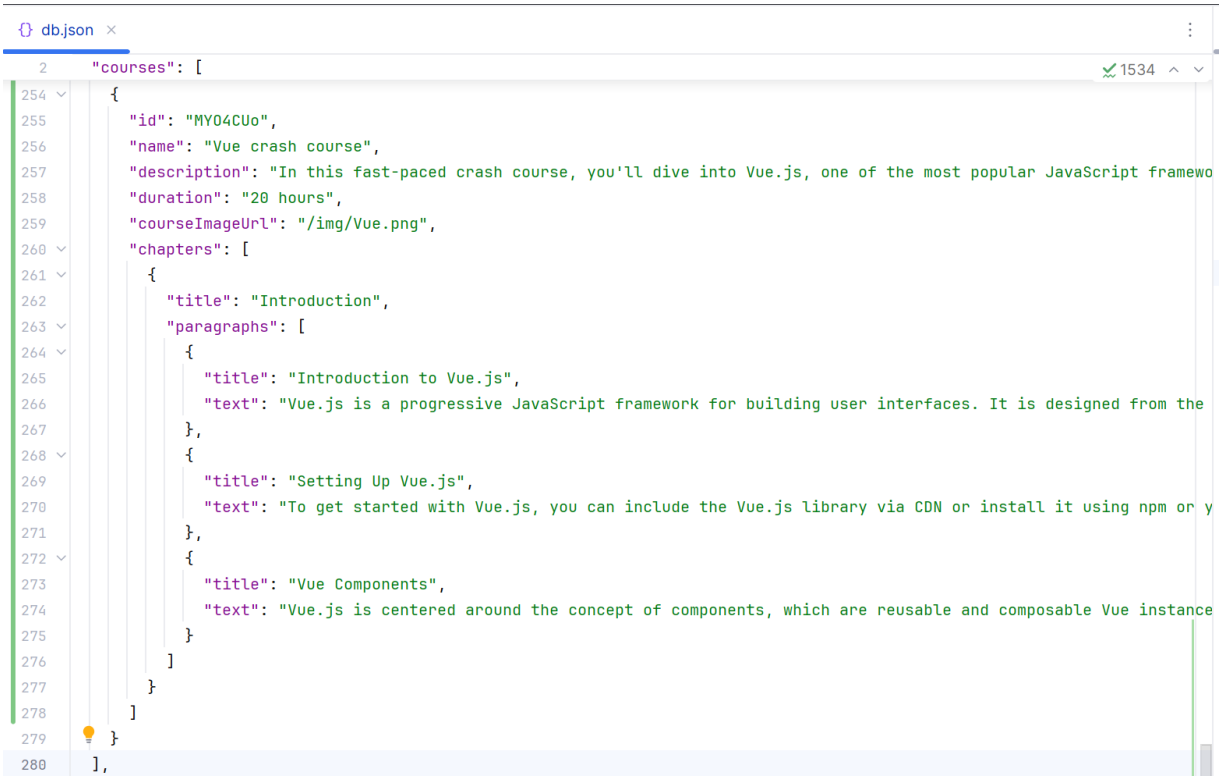


Рисунок 3.8 – Запис збереженого курсу у базі

На рисунку 3.9 продемонстровано розміщення збереженого курсу на головній сторінці вебзастосунку.

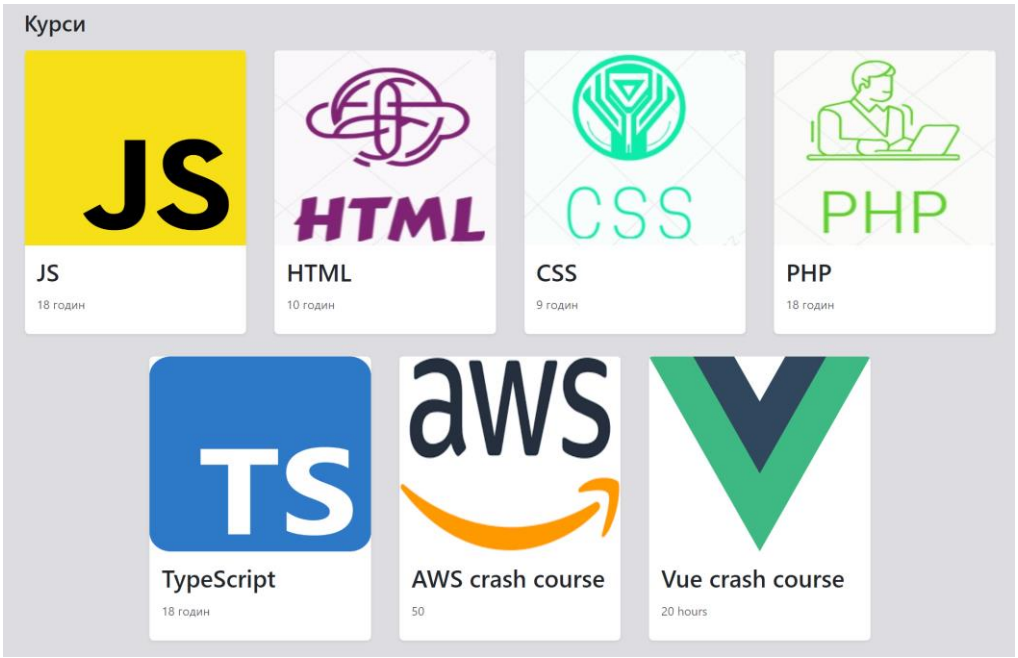


Рисунок 3.9 – Вигляд доданого курсу з головної сторінки

3.4 Розробка модуля для проходження тестування

Модуль для проходження тестування є ключовим компонентом освітніх платформ, онлайн курсів або систем оцінювання. Він дозволяє користувачам пройти тест, який містить набір питань або завдань, призначених для перевірки їх знань або навичок у певній області.

Компонент TestForm.vue є основним і відображає список питань, кожне з яких представлено окремим компонентом Question.vue. Після завершення тестування, результати відображаються за допомогою компоненту Result.vue.

Методи:

- FetchQuestions(). Метод викликається для отримання списку питань з сервера. Використовує Axios для відправлення GET запиту до API.

- SubmitAnswers(). Метод викликається при відправленні відповідей на сервер. Використовує Axios для відправлення POST запиту з відповідями користувача.

- CalculateScore(). Метод обчислює коефіцієнт результату на основі правильних відповідей. Реалізація цього методу залежить від логіки оцінювання конкретного тесту.

Використання Axios для відправлення відповідей на сервер (axios.post) та отримання питань (axios.get). API надає endpoints для отримання списку питань та відправлення відповідей.

Користувач запускає тест, викликуючи метод fetchQuestions() для завантаження питань. Під час проходження тесту користувач відповідає на питання, використовуючи поля для відповідей в компоненті Question.vue. Після завершення тесту користувач відправляє свої відповіді, викликуючи метод submitAnswers(), який відправляє дані на сервер. Сервер обробляє відповіді, обчислює бал та відправляє результати назад клієнту, які потім відображаються в компоненті Result.vue.

Цей модуль забезпечує гнучке та адаптивне середовище для проведення тестування, дозволяючи користувачам ефективно моніторити свій прогрес та отримувати відгук про рівень своїх знань або навичок.

Блок-схема процесу проходження тестування наведена на рисунку 3.10

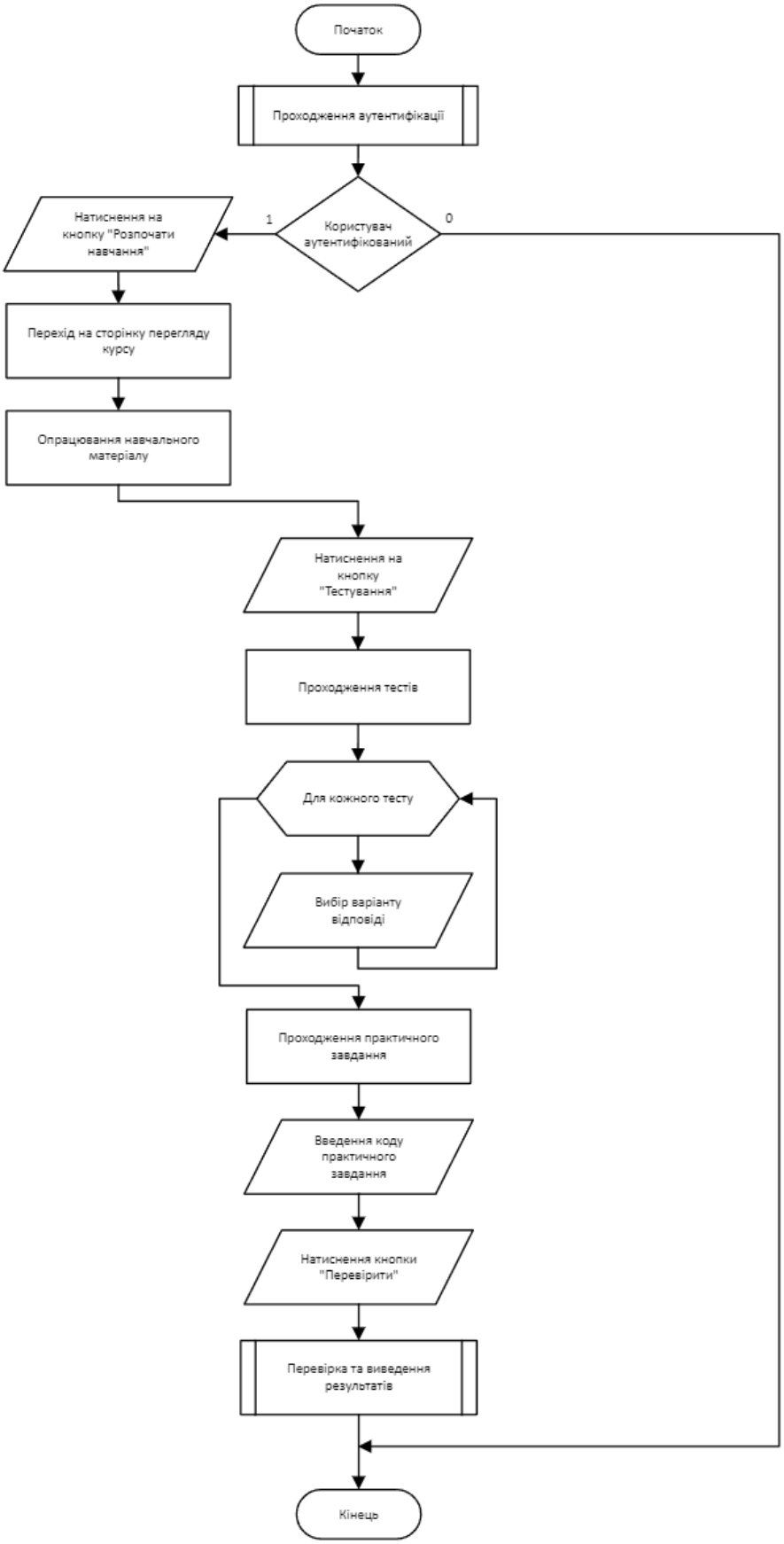


Рисунок 3.10 – Робота модуля проходження тестування

3.5 Розробка структури інтерфейсу системи

Структура інтерфейсу програми визначається тим, як користувач може взаємодіяти з програмою, як вона організована і які функції доступні. Основна мета структури інтерфейсу – забезпечити зручність і ефективність користування програмою, роблячи її функції доступними і зрозумілими для користувача.

Наведемо деякі рекомендації щодо розробки структури інтерфейсу:

1. Інтерфейс має бути простим і зрозумілим для користувача. Елементи керування повинні бути легко впізнаваними та легкодоступними.
2. Функції програми повинні бути логічно згруповані та організовані. Наприклад, меню повинно бути розміщене так, щоб користувач міг легко знайти потрібну опцію.
3. Елементи інтерфейсу повинні бути консистентними у всій програмі. Наприклад, використання однакових символів для позначення подібних дій або функцій.
4. Користувач повинен легко знаходити доступ до основних функцій програми без зайвих зусиль.
5. Інтерфейс повинен бути призначений для цільової аудиторії та враховувати їхні потреби.
6. Надання користувачеві інформації про те, як користуватися програмою, а також підказок у випадку виникнення проблем.
7. Інтерфейс має бути адаптований до різних пристроїв, таких як комп'ютери, планшети або смартфони.

При першому відвідуванні системи, користувачеві необхідно пройти процедуру реєстрації, щоб створити особистий обліковий запис. Це дає можливість отримати доступ до навчальних курсів та відстежувати власний прогрес. Для реєстрації, користувач відкриває спеціальне вікно, де заповнює необхідні поля з особистою інформацією. Після успішного створення облікового запису, користувач може використовувати систему повноцінно, зберігаючи свої досягнення та прогрес у навчанні.

Структура інтерфейсу вікна реєстрації наведена на рисунку 3.11.

Елементи вікна реєстрації:

1. Логотип системи.
2. Поле пошуку курсів.
3. Назва системи.
4. Поле для введення імені.
5. Поле для електронної адреси.
6. Поле для паролю.
7. Кнопка «Увійти» для тих користувачів, що вже мають аккаунт.
8. Кнопка «Створити аккаунт».

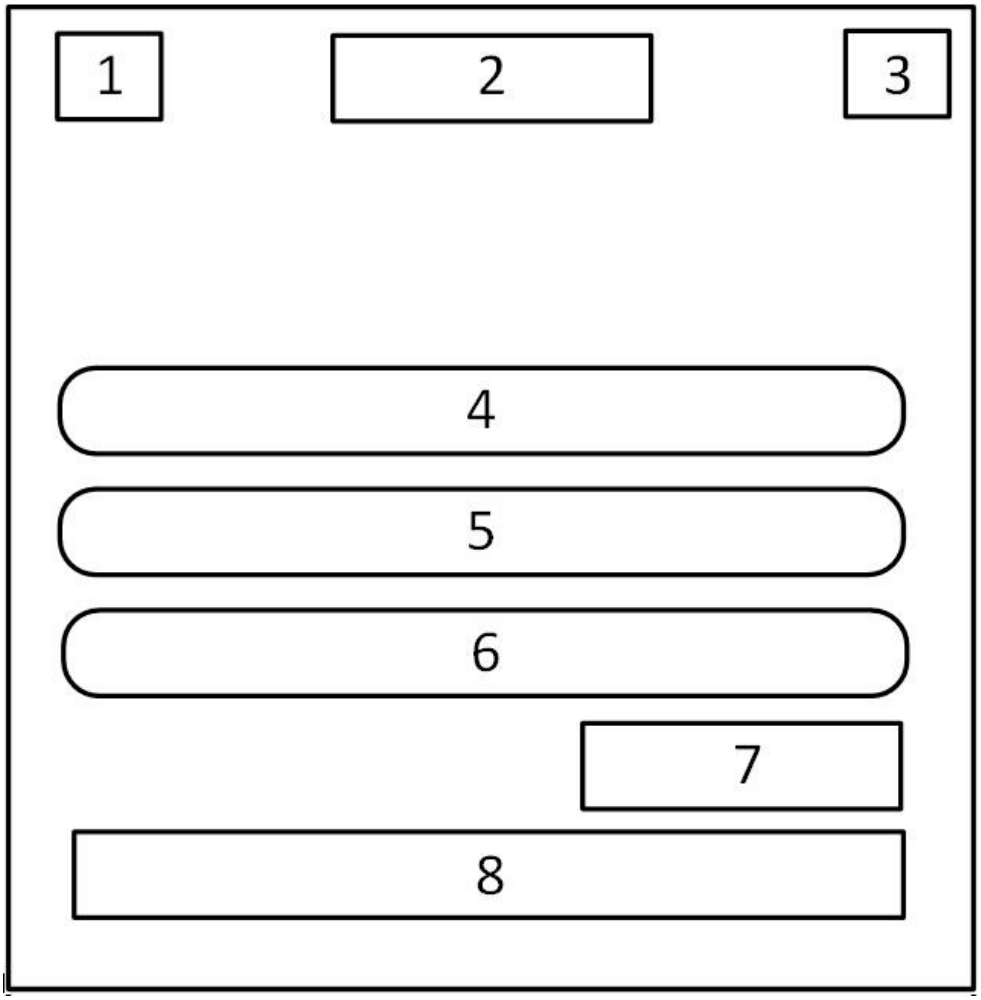


Рисунок 3.11 – Структурна схема вікна реєстрації

Вікно авторизації в системі, яка не використовує зовнішні сервіси для входу, фокусується на наданні користувачам простоти та безпеки при введенні своїх

унікальних даних — електронної пошти та паролю. Цей метод авторизації є основою для забезпечення безпечного доступу до особистих даних користувача. На рисунку 3.11 зображено структурну схему вікна авторизації користувача.

Елементи вікна авторизації:

1. Логотип системи.
2. Поле пошуку курсів.
3. Назва системи.
4. Поле для електронної адреси.
5. Поле для пароля.
6. Кнопка «Увійти».

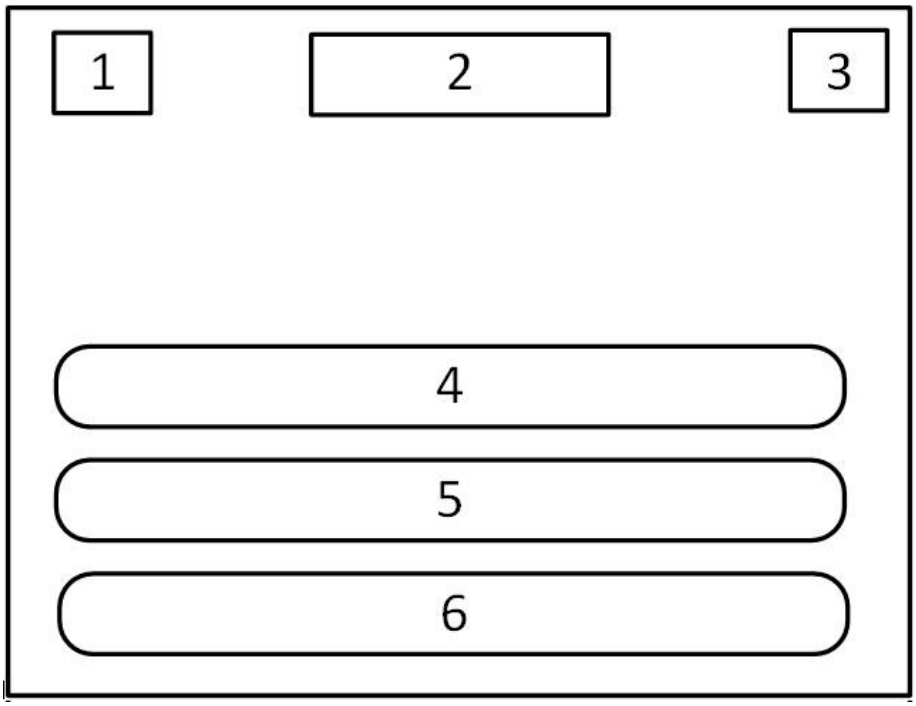


Рисунок 3.12 – Структурна схема вікна авторизації

Домашня сторінка системи не лише надає актуальну інформацію про доступні курси, але й дозволяє користувачам легко орієнтуватися в різноманітних можливостях для самовираження та розвитку. Вона також включає персоналізовані рекомендації, які допомагають кожному знайти те, що найбільше відповідає його індивідуальним потребам і цілям. Це робить процес вибору курсів більш зручним і ефективним, забезпечуючи користувачам можливість швидко

знайти ті навчальні матеріали, які вони шукають. Структурну схема домашньої сторінки наведена на рисунку 3.12.

Елементи домашньої сторінки системи:

1. Логотип системи
2. Поле пошуку курсів.
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Банер.

6-11. Рекомендовані курси.

12. Футер вебсайту.

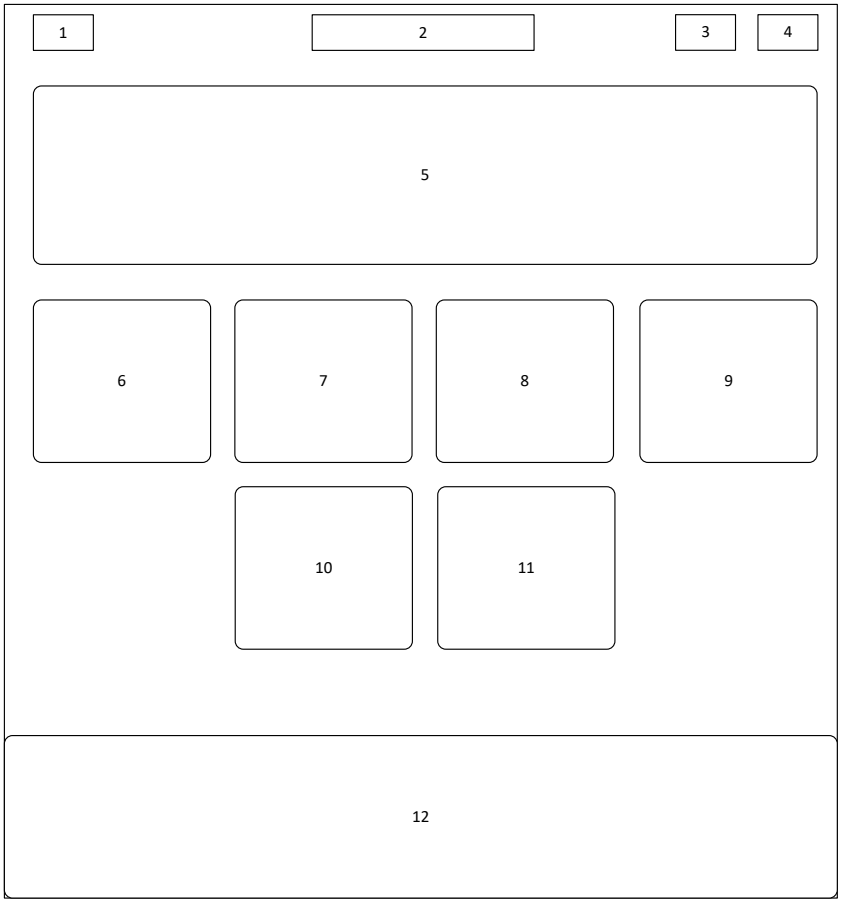


Рисунок 3.13 – Структурна схема головного вікна системи

Вікно пошуку курсів дозволяє користувачам знаходити курси за своїми потребами. Для цього, проводиться пошук в базі даних курсів за ключовими словами запиту користувача та демонструється список курсів, що підпадають під

цей запит. Користувач може фільтрувати курси за багатьма критеріями та сортувати список курсів залежно від своїх потреб.

Структурну схему вікна пошуку курсів наведено на рисунку 3.14.

Елементи вікна пошуку:

1. Логотип системи.
2. Рядок пошуку
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Рядок зі змістом запиту та кількістю знайдених курсів.

6-9. Знайдені курси.

10. Футер вебсайту.

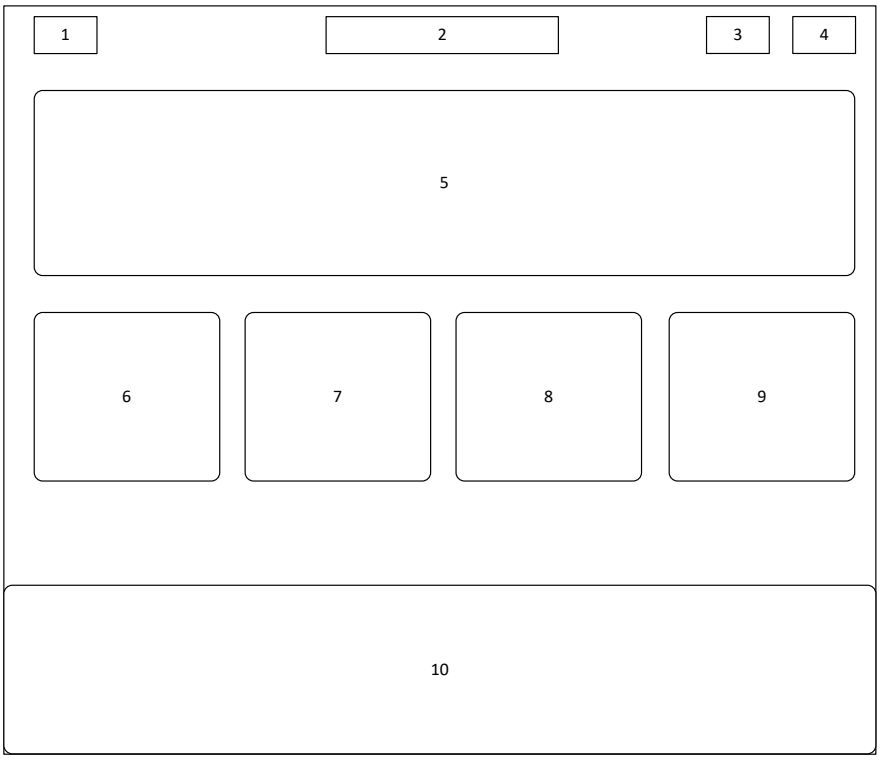


Рисунок 3.14 – Структурна схема вікна пошуку курсів

Після вибору певного курсу для перегляду, користувачу спершу презентується вікно перегляду інформації про курс, структурну схему якого наведено на рисунку 3.15. Кожен курс представлений у вигляді карточки з назвою, коротким описом та кнопкою "Розпочати Навчання". При натисканні на цю кнопку,

користувач отримує доступ до повного курсу, де може знайти всі необхідні матеріали для вивчення. Крім цього, міститься інформація про загальну тривалість курсу, яка допомагає користувачу оцінити, скільки часу йому знадобиться для завершення курсу.

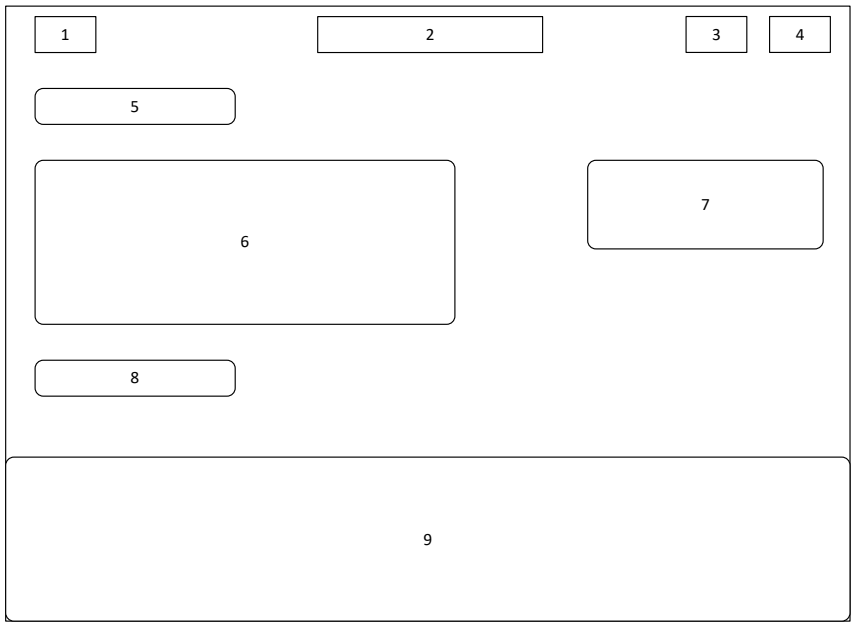


Рисунок 3.15 – Структурна схема вікна перегляду загальної інформації про курс

Елементи вікна перегляду інформації про курс:

1. Логотип системи.
2. Рядок пошуку
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Назва курсу.
6. Секція з описом курсу.
7. Логотип до курсу.
8. Кнопка «Розпочати навчання».
9. Футер вебсайту.

Вікно перегляду вебкурсу надає доступ до перегляду матеріалів курсу та перемикання між різними його розділами та параграфами. Структурна схема вікна перегляду курсу наведено на рисунку 3.16.

Елементи вікна перегляду курсу:

1. Логотип системи.
2. Рядок пошуку
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Назва курсу.
6. Секція з описом курсу.
7. Логотип до курсу.
8. Кнопка «Розпочати навчання».
9. Назва розділу
10-13. Назва теми
14. Футер вебсайту.

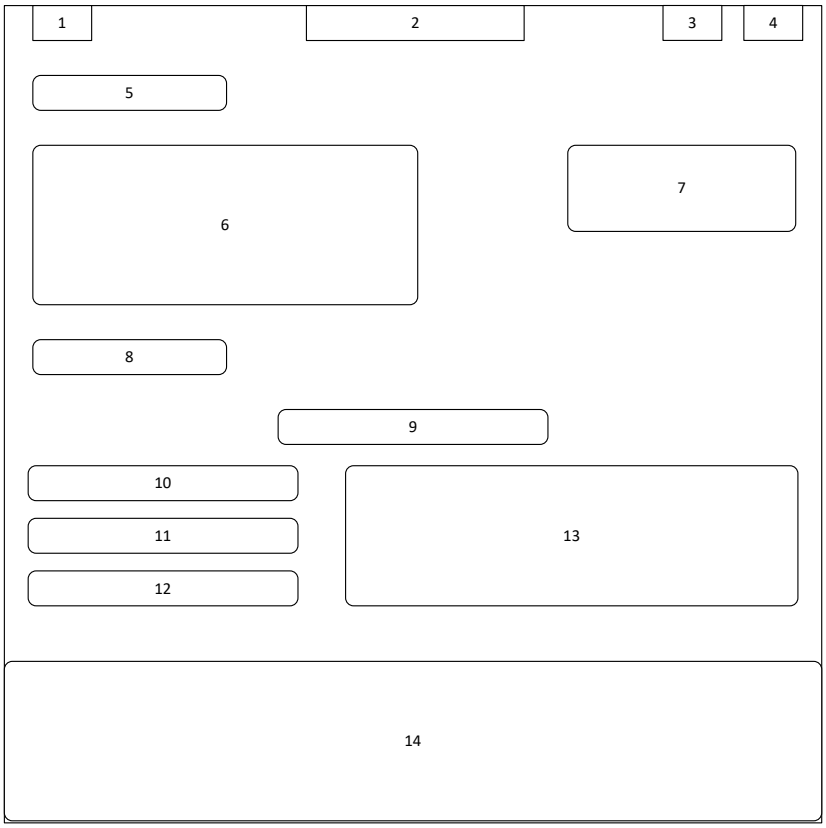


Рисунок 3.16 – Структурна схема вікна перегляду курсу

Вікно тестування дозволяє перевірити рівень знань та рівень їх засвоєння студентами – слухачами курсу. Вони розробляються викладачами курсів.

Структурна схема вікна тестування наведена на рисунку 3.17.

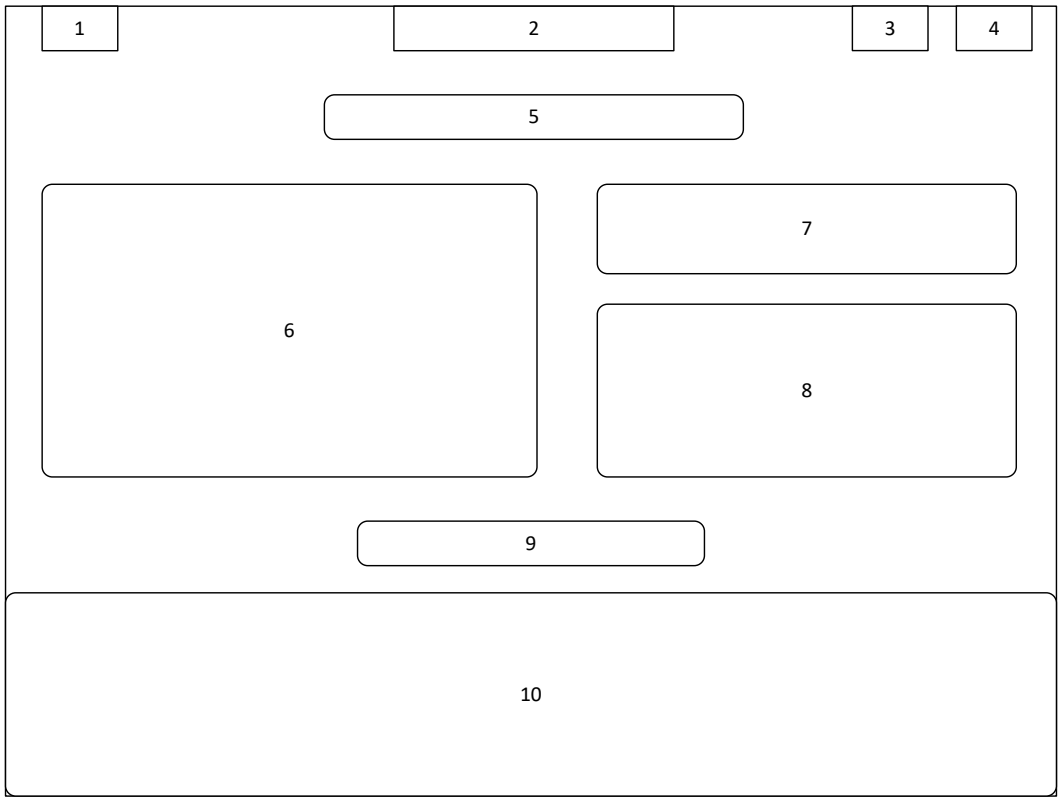


Рисунок 3.17 – Структурна схема вікна тестування

Елементи вікна тестування:

1. Логотип системи.
2. Пошукове поле.
3. Назва тесту.
4. Секція тестування.
5. Секція практичного завдання.
6. Область для набору коду до практичного завдання.
7. Кнопка «Перевірити».
8. Футер системи.

Вікно створення курсів розроблене для експертів, що бажають поділитися своїми знаннями з іншими, менш кваліфікованими розробниками. Воно дозволяє створювати курси, додавати до них відео та опис курсів.

Структурна схема вікна створення курсів наведена на рисунку 3.18.

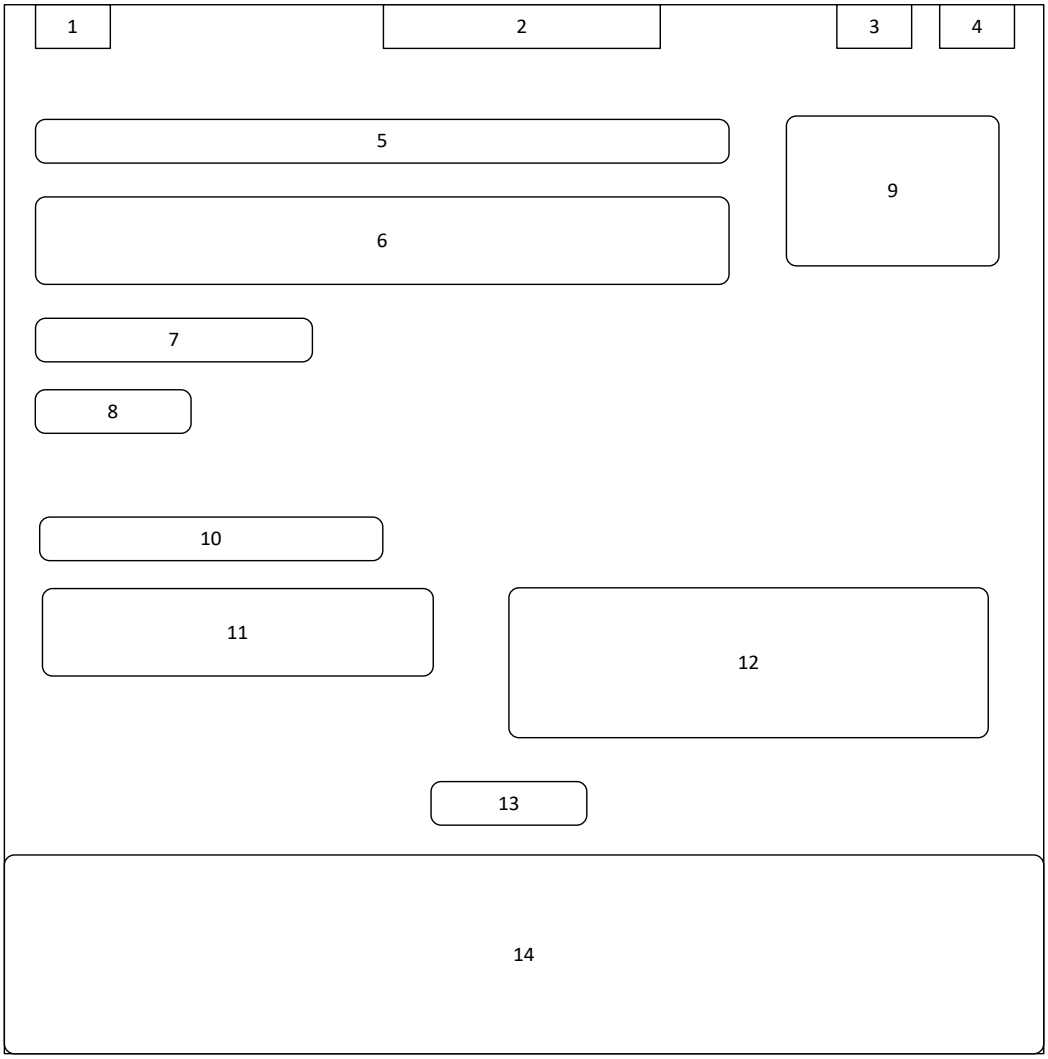


Рисунок 3.18 – Структурна схема вікна створення курсу

Елементи вікна створення курсів:

1. Логотип системи.
2. Рядок пошуку.
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Назва курсу.
6. Панель форматування тексту для опису.
7. Поле для опису кількості годин.
8. Кнопка «Додати курс».
9. Логотип курсу.

10. Поле для назви розділу курсу.
11. Поле для параграфів розділів.
12. Поле для опису розділу.
13. Кнопка «Додати розділ».
14. Футер вебсайту.

Отже, було розроблено структуру інтерфейсу навчальної системи для вивчення основ вебпрограмування для початківців.

Вікно перегляду інформації користувача дозволяє користувачу переглядати персональну інформацію, відстежувати перелік курсів, які він проходить, та прогрес виконання.

Структурна схема вікна перегляду користувача наведена на рисунку 3.19.

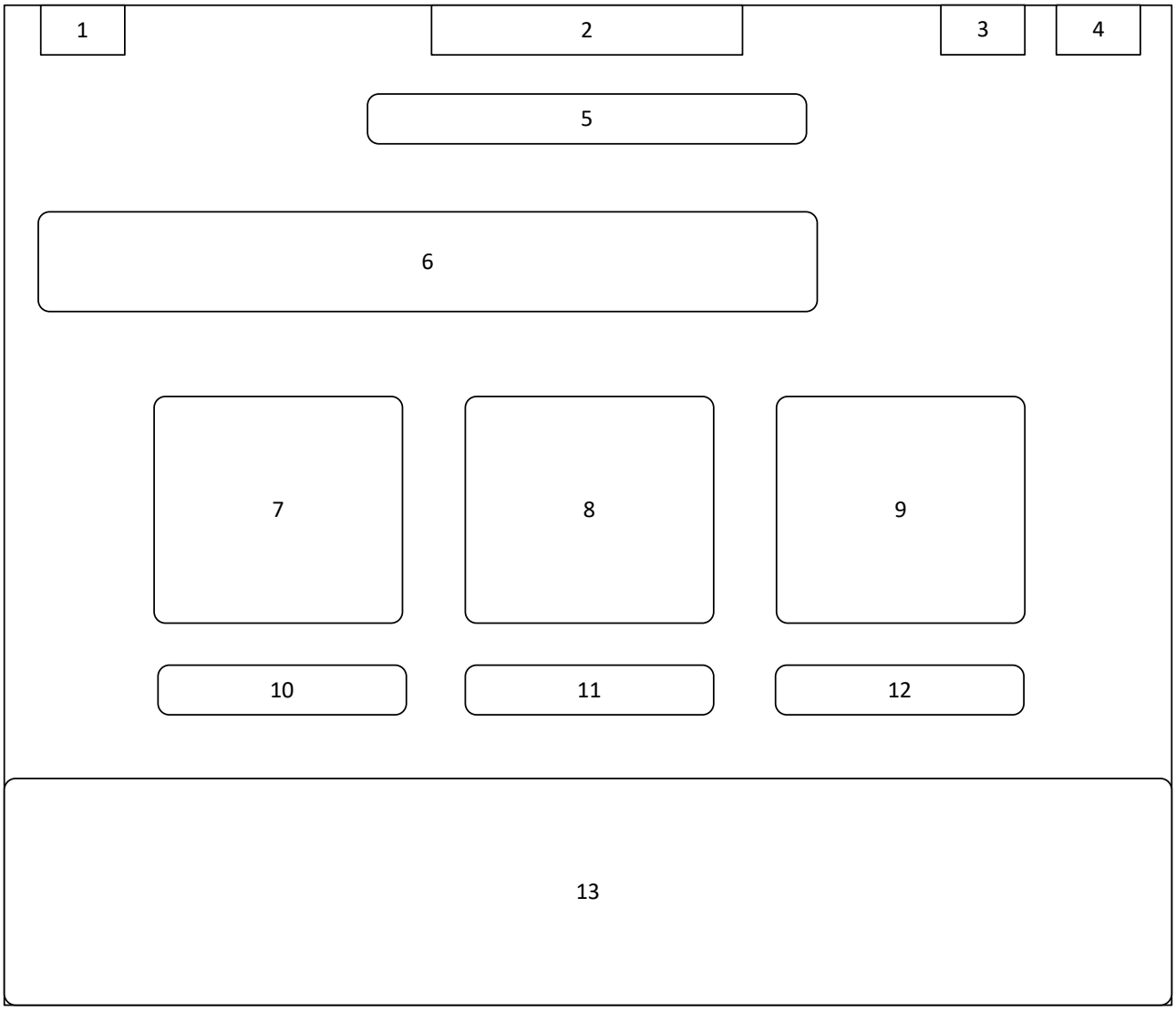


Рисунок 3.19 – Структурна схема вікна перегляду користувача

Елементи вікна перегляду користувача:

1. Логотип системи.
2. Пошуковий рядок.
3. Кнопка «Профіль Користувача».
4. Кнопка «Вихід».
5. Заголовок вікна перегляду користувача
6. Інформація користувача.
7-9. Курси користувача.
10-12. Прогрес проходження курсів.
13.Футер.

3.6 Висновки

У третьому розділі було проведено варіантний аналіз та обґрунтування вибору мови програмування, вибору середовища розробки та вибору засобів для реалізації бази даних. Було описано кроки реалізації розробки навчальної системи для вивчення основ вебпрограмування для початківців. Розроблено структуру інтерфейсу навчальної системи для вивчення основ вебпрограмування для початківців.

4 ТЕСТУВАННЯ НАВЧАЛЬНОЇ СИСТЕМИ

4.1 Тестування навчальної системи для вивчення основ вебпрограмування для початківців

Тестування вебсистем – це важливий процес, який дозволяє перевірити, чи вебсайт або вебдодаток відповідає всім необхідним вимогам та функціонує належним чином. Основні етапи тестування вебсистем включають:

1. Планування тестування:

- Визначення цілей та scope тестування.
- Складання плану тестування.
- Вибір методів та інструментів тестування.

2. Підготовка до тестування:

- Налаштування тестового середовища.
- Створення тестових сценаріїв та даних.
- Розробка автоматизованих тестів.

3. Виконання тестування:

- Запуск функціональних тестів (перевірка основних функцій вебдодатку).
- Проведення юзабіліті-тестів (аналіз зручності використання сайту).
- Проведення тестів продуктивності (оцінка швидкодії, стійкості до навантаження).
- Перевірка сумісності з різними браузерами та пристроями.
- Виявлення та реєстрація дефектів.

4. Аналіз результатів та звітування:

- Аналіз та приоритизація виявлених дефектів.
- Складання звітів про результати тестування.
- Надання рекомендацій щодо покращення вебсистеми.

Під час тестування можуть використовуватися різноманітні методи, такі як: модульне тестування, інтеграційне тестування, системне тестування, smoke-тестування, регресивне тестування тощо. Також застосовуються автоматизовані інструменти тестування, які допомагають прискорити та покращити якість

перевірок.

На рисунку 4.1 зображено тестування головної сторінки системи. На ній відображаються доступні рекомендовані системою курси залежно від вподобань та попередніх переглядів користувача.

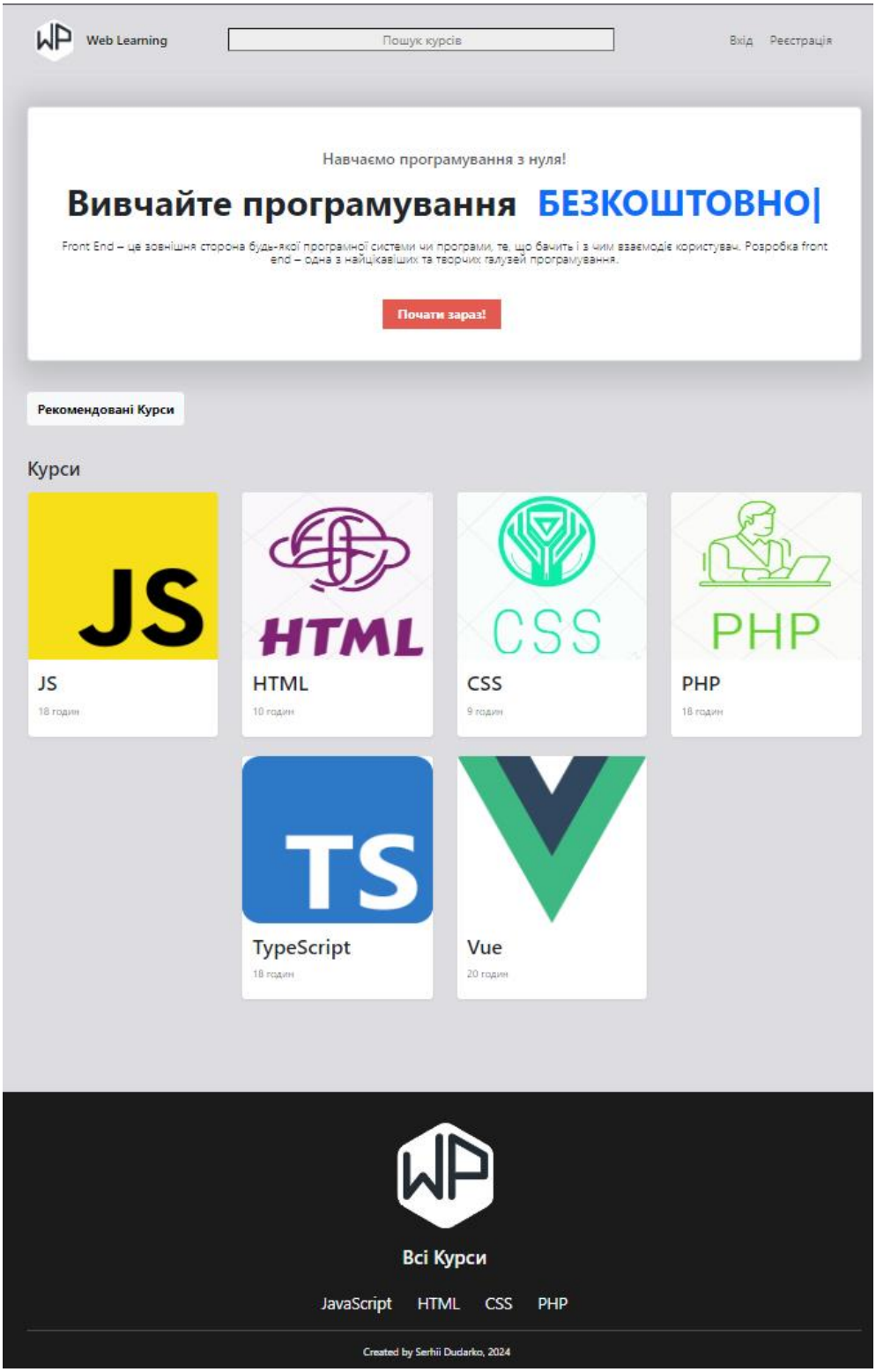


Рисунок 4.1 – Тестування головної сторінки системи

На рисунку 4.2 зображено тестування модуля пошуку. В цьому прикладі, було здійснено пошук за запитом «Веб програмування». В результаті, система демонструє перелік курсів, пов’язаних з розробкою користувацьких інтерфейсів.

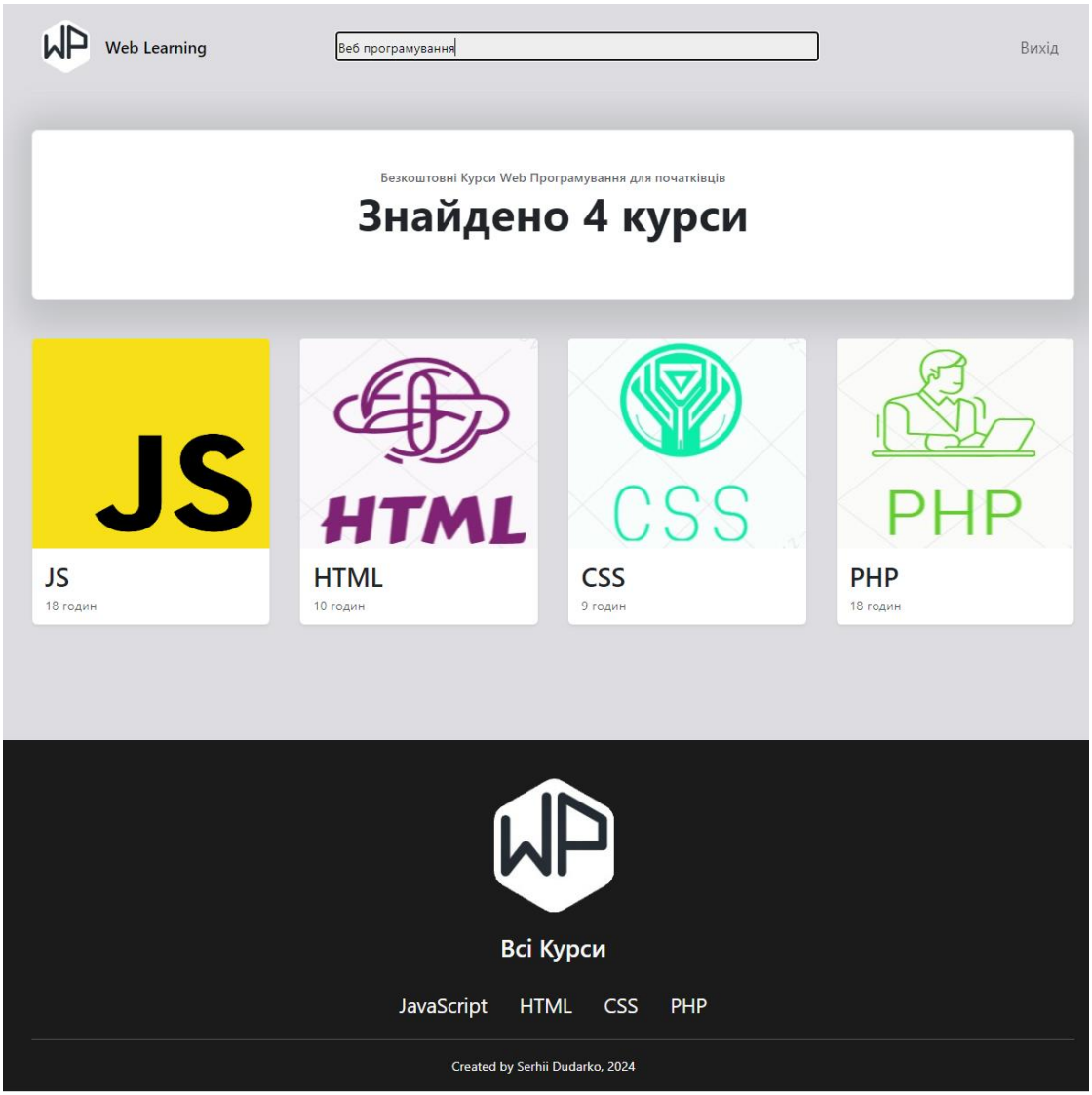


Рисунок 4.2 – Тестування рядка пошуку курсів

Оберемо один курс зі списку. При натисканні на курс, користувач переходить у вікно перегляду курсу. В ньому відображено назву, детальний опис, кнопку для переходу до навчання.

Результат тестування вкладки перегляду інформації про курс наведено на рисунку 4.3.

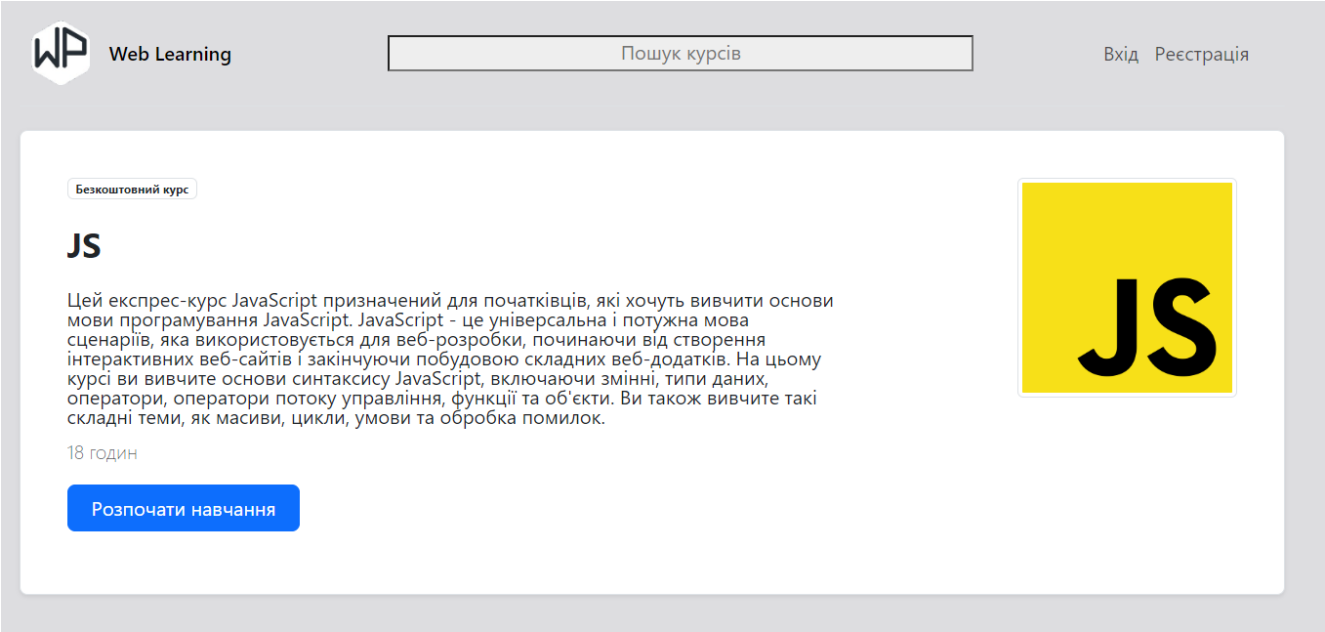


Рисунок 4.3 – Перегляд інформації про курс

Після авторизації користувачем, стають доступними всі матеріали з цього курсу. Стає доступним функціонал для перемикання між темами та перегляд їхнього переліку. Результат тестування вкладки перегляду вибраного курсу наведено на рисунку 4.4.

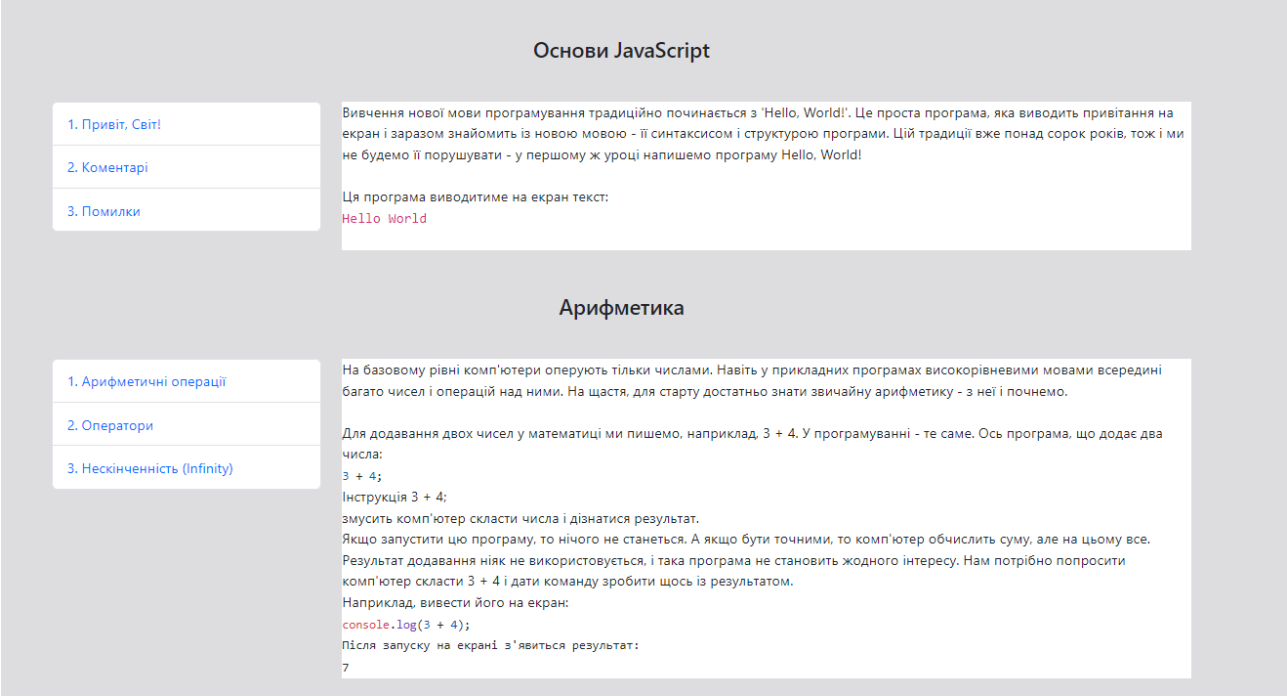


Рисунок 4.4 – Вкладка вибраного курсу

Після проходження курсу, користувач може пройти перевірку отриманих ним навичок. Для цього йому доступне окреме вікно, в лівій частині якого можна пройти тестування з кількома варіантами відповіді, а також виконати практичне завдання, яке передбачає написання коду для вирішення певної задачі. Вікно тестування наведено на рисунку 4.5.

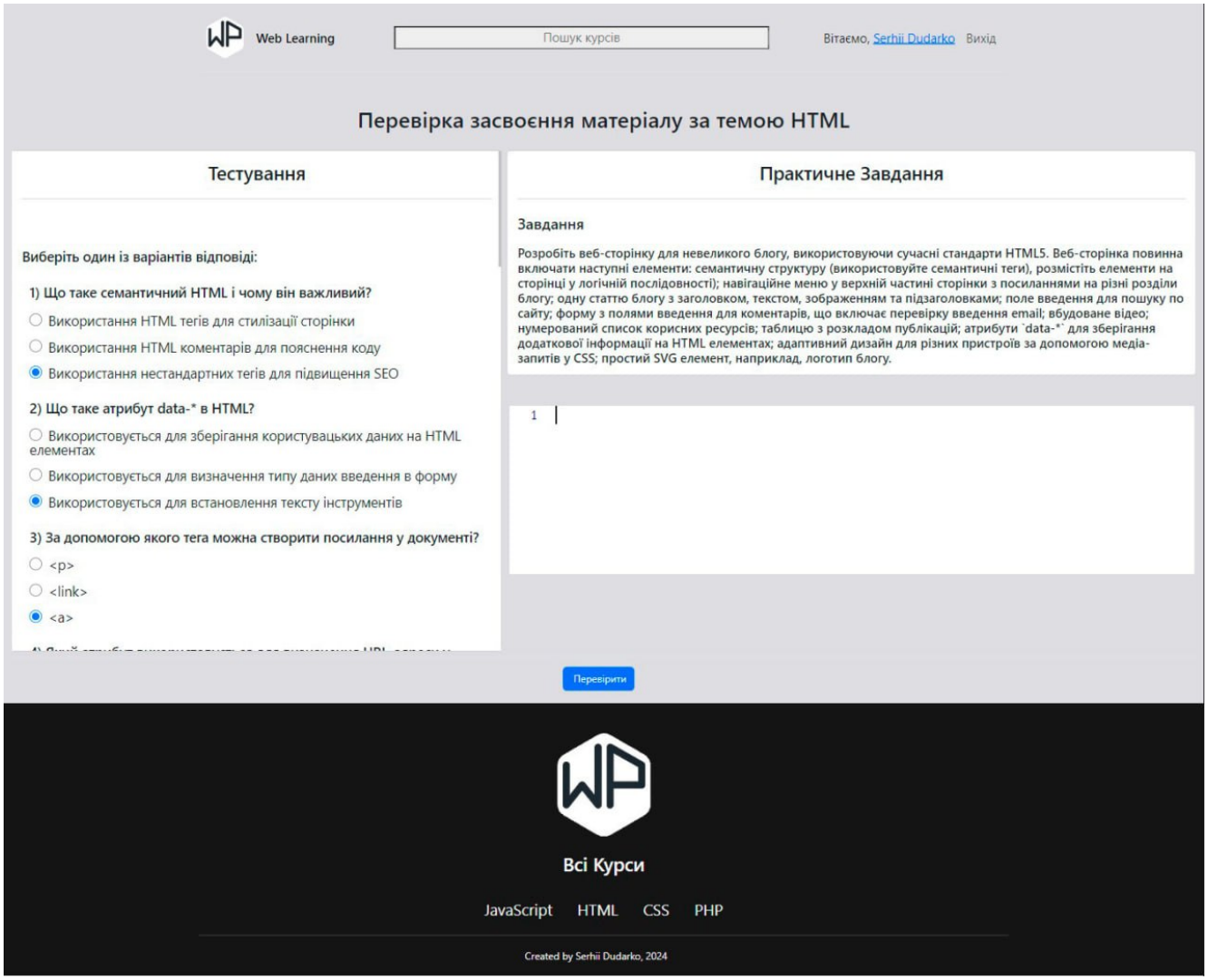


Рисунок 4.5 – Вікно тестування

Вікно додавання курсу адміністратором дозволяє створювати нові курси в системі. Для цього адміністратору доступні відповідні поля для введення назви курсу, його опису, тривалості проходження курсу, логотипу курсу.

Окрім цього, адміністратор має можливість створювати розділи та додавати параграфи для розбиття курсу на секції.

Вікно додавання курсу адміністратором наведено на рисунку 4.6.

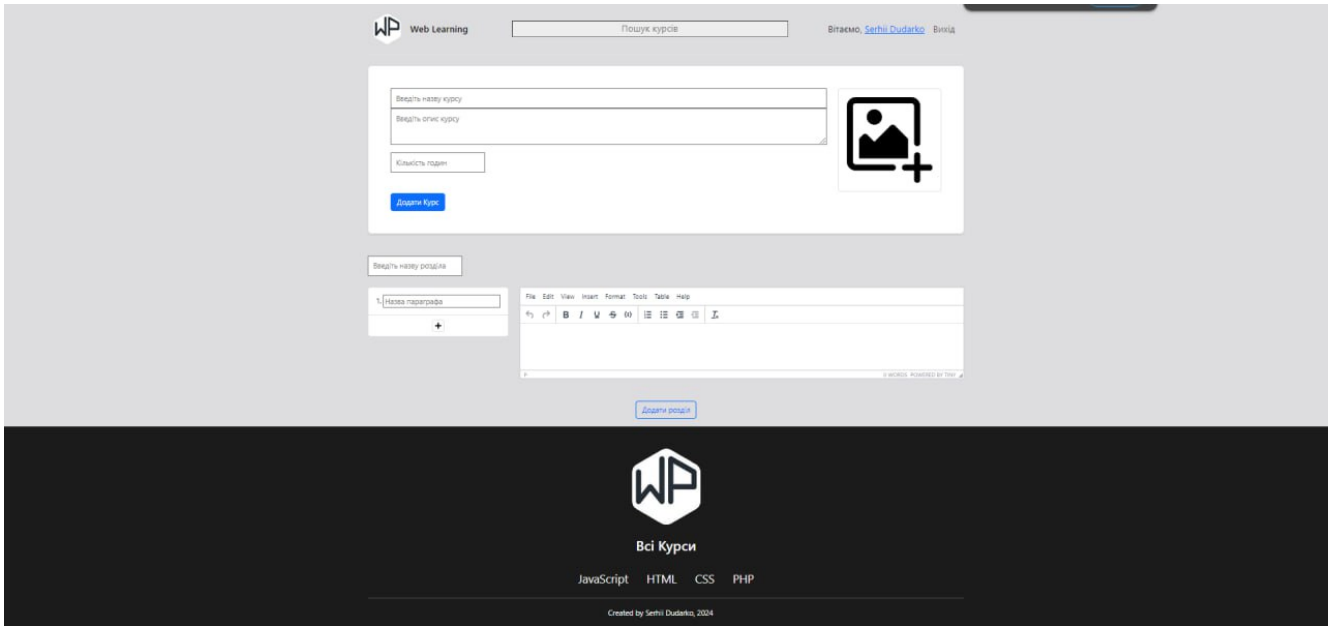


Рисунок 4.6 – Вікно додавання курсу адміністратором

Вікно редагування курсу є схожим на вікно додавання курсу. В нього завантажується вже додана раніше інформація про курс, яку адміністратор може змінити за бажанням. Зображення вікна редагування курсу наведено на рисунку 4.7.

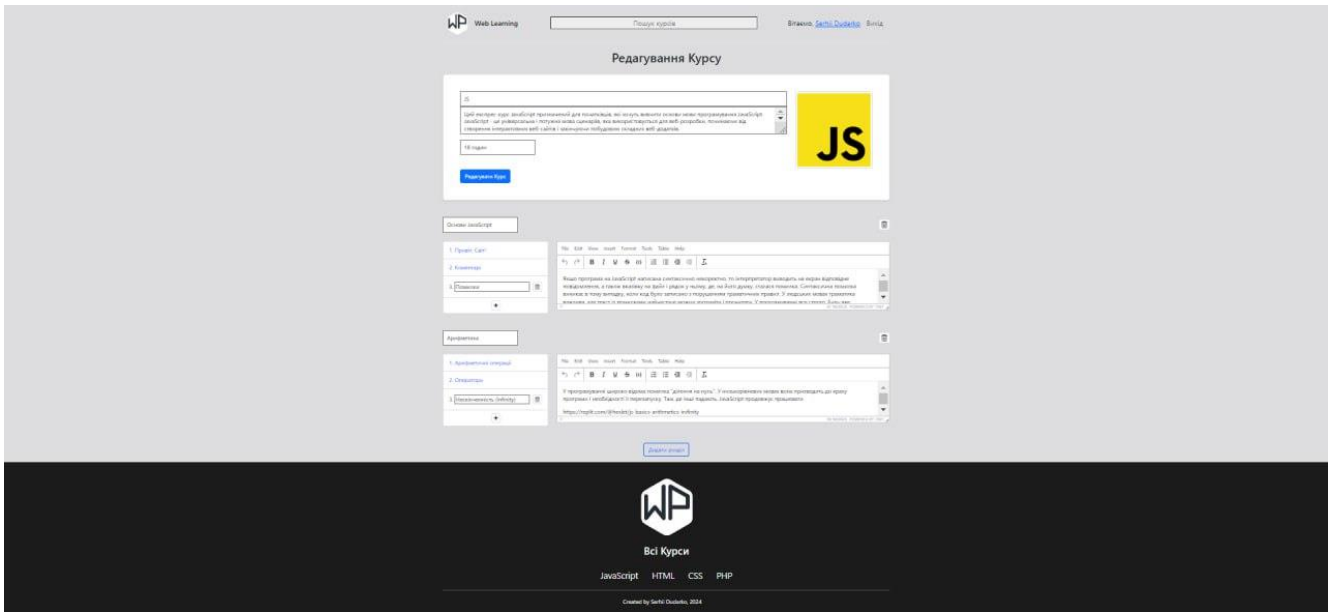


Рисунок 4.7 – Вікно редагування курсу адміністратором

Вікно інформації про користувача дозволяє переглядати йому персональну інформацію, що міститься в системі, його ролі, та курси, що він проходить. Окрім цього, міститься прогрес проходження курсів. Зображення вікна інформації про користувача наведено на рисунку 4.8.

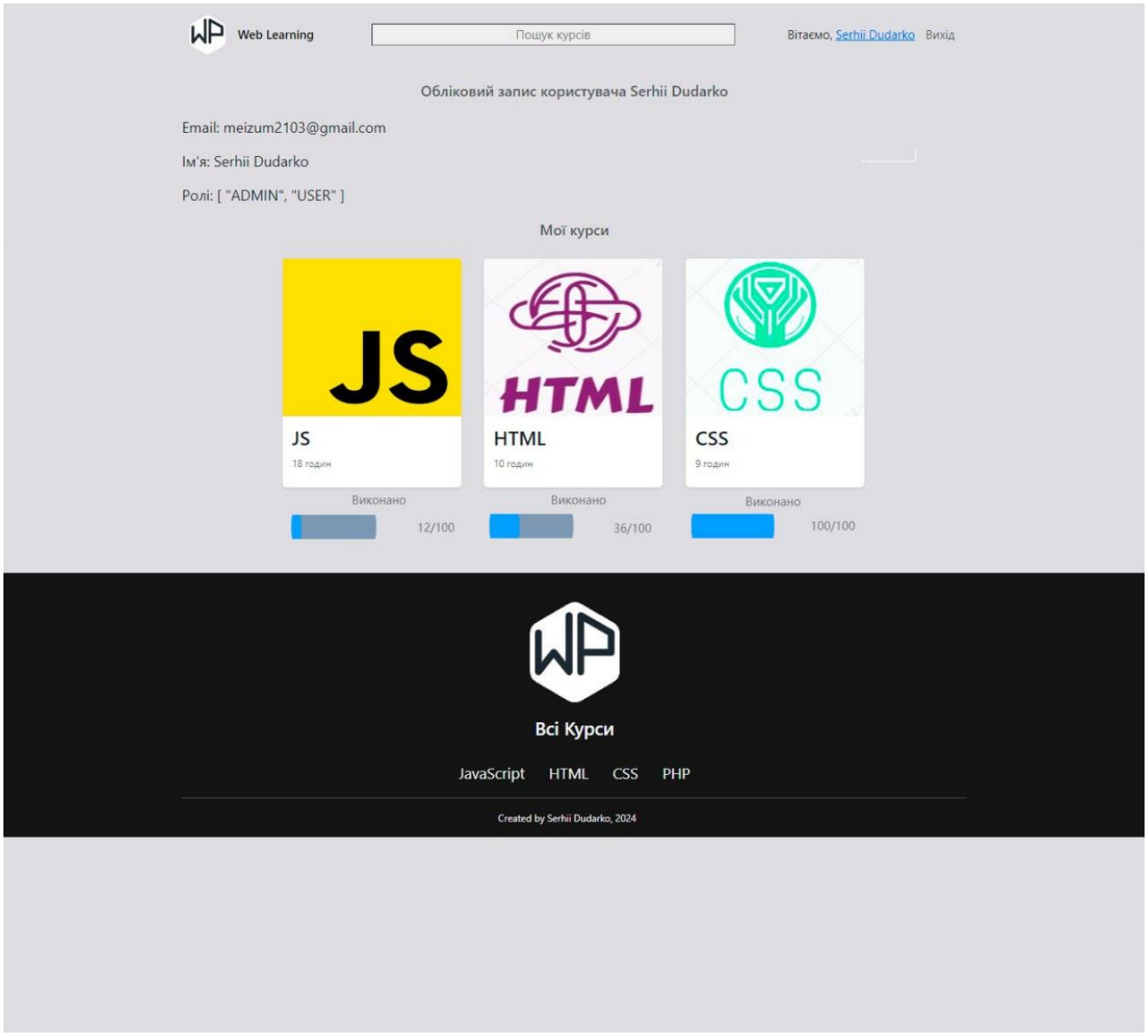


Рисунок 4.8 - Інформація про користувача

Таким чином, було проведено тестування навчальної системи для вивчення основ вебпрограмування, продемонстровано її функціонал. Розроблена вебсистема виконує поставлені перед нею на початку розробки завдання.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску системи, які можна знайти у таблицях 4.1 та 4.2

Таблиця 4.1 – Мінімальні конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i3 U6100 2.4 GHz або Intel Pentium N4200 2.5 GHz
Оперативна пам'ять	4 GB
Пропускна здатність мережі	50 Mbps
Вільного місця на диску	150 Мб

Таблиця 4.2 – Рекомендовані конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i5-1135G7 4.2 GHz або AMD Ryzen 5500U 4.0 GHz
Оперативна пам'ять	6 GB
Пропускна здатність мережі	150 Mbps
Вільного місця на диску	200 Мб

Інструкція користувача навчальної системи для вивчення основ вебпрограмування для початківців:

1. Користувач переходить у систему за відповідним посиланням.

2. Користувач реєструється у системі, використовуючи один з доступних способів – через електронну пошту.

3. Якщо користувач уже зареєстрований, він переходить у вкладку авторизації та використовує для цього той же спосіб, що і при реєстрації. В разі правильно введених даних, він отримує доступ до свого аккаунту в системі, якщо ні – система виводить помилку про неправильно введені дані.

4. Після авторизації, користувач переходить у головне вікно системи. На ньому, йому пропонується переглянути певних перелік курсів. Натиснувши на

нього, йому відкривається вкладка детальної інформації про цей курс.

5. Користувач, окрім головної сторінки, може знайти курс за цікавою йому тематикою, увівши відповідний запит у поле пошуку, що знаходиться у верхній частині екрану на головній сторінці. Після підтвердження запиту для пошуку, відкривається нове вікно, у якому користувачеві відображаються курси за ключовими словами.

6. У верхній правій частині усіх сторінок системи доступна кнопка для перегляду аккаунту. Там користувач може переглянути відомості про свої облікові дані та прогрес по вивченню ним курсів.

Інструкція для адміністраторів навчальної системи описує процес керування курсами. Адміністратори мають такий функціонал у навчальній системі:.

1. Адміністратор може створювати нові курси в системі, заповнюючи відповідну форму додавання курсу. Під час додавання курсу адміністратор вказує назву курсу, опис, тривалість проходження курсу та інші важливі характеристики. Крім того, адміністратор має можливість додавати розділи та параграфи для структурування курсу.

2. Адміністратор може редагувати інформацію про існуючі курси, таку як назва, опис, тривалість та інші характеристики. Якщо змінюється структура курсу, наприклад, додаються нові розділи або параграфи, адміністратор також може внести необхідні зміни.

4.3 Висновки

У четвертому розділі було описано особливості тестування навчальних систем, проведено тестування розробленої навчальної системи для вивчення вебпрограмування для початківців. Розроблено інструкцію користувача системи.

ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи, було розроблено навчальну систему для вивчення основ програмування для початківців. Даний програмний застосунок забезпечує можливість вивчення програмування початківцями шляхом перегляду курсів від інструкторів з можливістю проходження тестування для перевірки отриманих знань та навичок.

Для розробки було використано мову програмування JavaScript, середовище розробки WebStorm та засоби AWS для аутентифікації користувача, бази даних та сховища даних.

У першому розділі, було проведено аналіз стану питання навчальної системи, порівняльний аналіз аналогів, аналіз методів розв'язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі розробки.

У другому розділі, було розроблено структуру навчальної системи для початківців, побудовано її модель, розроблено алгоритми роботи системи та структурні схеми інтерфейсу системи.

У третьому розділі, було проведено варіантний аналіз та обґрунтування вибору мови програмування, середовища розробки та засобів для реалізації бази даних. Розроблено навчальну систему.

У четвертому розділі, було проведено аналіз методів тестування, проведено тестування розробленої навчальної системи. Тестування довело її працездатність та коректність роботи.

Відповідно до поставлених задач було:

- проведено аналіз існуючих навчальних систем для вивчення вебпрограмування, визначено їхні переваги та недоліки;

- розроблено метод, модель та алгоритми навчальної системи, що враховує потреби та вимоги студентів, а також сучасні тенденції у веброзробці;

- реалізовано базовий функціонал вебсистеми, який включає:

1) реєстрацію/ аутентифікацію користувачів

2) перегляд навчальний матеріалів;
3) тестування за темами курсів та інтерактивну область для написання коду;
4) розміщування нових курсів та розробка їх наповнення;

- проведено тестування вебсистеми, що підтвердило її працездатність і коректність роботи.

Розроблена система виконує поставлені перед нею на початку розробки задачі.

Бакалаврська дипломна робота оформлена відповідно до методичних вказівок [21].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Online courses [Електронний ресурс] – Режим доступу до ресурсу: https://www.reed.co.uk/courses/online

2. Дударко С.Є. Розробка навчальної системи для вивчення основ вебпрограмування/В. В. Войтко, Г. О. Черноволик, Л. М. Круподьорова, А. В. Денисюк, С. Є. Дударко // Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного унівреситету (2024) », Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20416/16965

3. Codeacademy [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/

4. Udacity [Електронний ресурс] – Режим доступу до ресурсу: https://www.udacity.com/

5. freeCodeCamp [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/

6. What is SOA? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/service-oriented-architecture/?nc1=h_ls

7. Amazon Cognito [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/cognito/?nc1=h_ls

8. DocumentDB[Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/documentdb/?nc1=h_ls

9. Amazon S3 [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/s3/?nc1=h_ls

10. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

11. PHP [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/

12. C# language documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/csharp/

13. NodeJS [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en

14. API [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/API

15. VueJS [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

16. Axios [Електронний ресурс] – Режим доступу до ресурсу: https://axios-http.com/docs/intro

17. Introducing JSON [Електронний ресурс] – Режим доступу до ресурсу: https://www.json.org/json-en.html

18. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/

19. HTTP status codes [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

20. HTTP POST [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

21. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Романюк О.Н.
"12" березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка навчальної системи для вивчення основ вебпрограмування»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____к.т.н., доц. В.В. Войтко
« 12 » березня 2024 р.

Виконав:
_____студент гр. 1ПІ-20б С.Є. Дударко
« 12 » березня 2024 р.

м. Вінниця – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка навчальної системи для вивчення основ вебпрограмування».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №80 від 11 березня 2024 р.

3. Мета та призначення розробки.

Метою роботи є підвищення навчальних можливостей вивчення вебпрограмування за рахунок розробки і використання спеціалізованої вебсистеми, яка допомагає у вивченні основ вебпрограмування.

Призначення роботи – розробка програмного продукту для вивчення осноов вебпрограмування.

4. Вихідні дані для проведення НДР

1. Online courses [Електронний ресурс] – Режим доступу до ресурсу: https://www.reed.co.uk/courses/online

2. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

3. WebStorm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/

4. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/

5. Технічні вимоги

Вхідні дані — підготовленні навчальні матеріали основ вебпрограмування.

Вихідні дані — збережені навчальні матеріали у вигляді курсів для вивчення основ вебпрограмування.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку вебсистем для вивчення основ вебпрограмування	14.03.24 – 20.03.24
2	Розробка методів, моделі та алгоритмів роботи програми	22.03.24 – 12.04.24
3	Розробка програмних модулів	13.04.24 – 13.05.24
4	Тестування навчальної системи	14.05.24 – 20.05.24
5	Оформлення матеріалів до захисту БДР	21.05.24 – 30.05.24

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Додаток Б – Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка навчальної системи для вивчення основ вебпрограмування.

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Войтко В.В.

Оригінальність	96.8%
Схожість	3.2%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Дударко С.Є.

Керівник роботи _____ Войтко В.В.

Додаток В – Лістинг програми

Лістинг файлу Application.vue

```
<template>
  <NavHeader />
  <router-view :key="$route.fullPath"/>
  <NavFooter />
</template>

<script>
import NavHeader from '@/components/parts/NavHeader.vue'
import NavFooter from "@/components/parts/NavFooter.vue";

export default {
  name: 'WebLearnApplication',
  components: {
    NavHeader: NavHeader,
    NavFooter: NavFooter,
  }
}
</script>

<style>

</style>
```

Лістинг файлу authorization.js

```
import axios from "axios";

export {
    isAdmin,
    isLoggedIn,
    base64UrlDecode,
    getAuthorizationUser,
    fetchAuthorizationUser
}

function extractJwtClaims(token) {
    let claims = token.split(".")[1];
    let user = JSON.parse(base64UrlDecode(claims));
    return user;
}

function isAdmin() {

    if (isLoggedIn.bind(this) ()) {
        let token = this.$store.getters.getAuthorization;
        console.log(typeof token)
        let user = extractJwtClaims(token);

        return user.roles.includes("ADMIN");
    }

    else {
```

```
        return false
    }
}

function isLoggedIn() {
    console.log(this.$store.getters.getAuthorization)
    return this.$store.getters.isAuthenticated;
}

function getAuthorizationUser() {
    console.log(this.$store.getters.getAuthorization)
    return this.$store.getters.getAuthorizationUser;
}

function fetchAuthorizationUser() {
    let authorization = this.$store.getters.getAuthorization;
    let claims = extractJwtClaims(authorization);
    axios.get("http://localhost:3000/users/" + claims.id, {
      headers: {
          "Authorization": "Bearer " + authorization
      },
    }).then(response => {
        let user = response.data;
        console.log(user);
        this.$store.commit('setAuthorizationUser', user);
    })
}

function base64UrlDecode(str) {
    while (str.length % 4 !== 0) {
        str += '=';
    }

    let base64 = str.replace(/-/g, '+').replace(/_/g, '/');

    let decoded = atob(base64);

    return decoded;
}
```

Лістинг файлу main.js

```
import {createApp} from 'vue'
import {createStore} from 'vuex'
import WebLearnApplication from './Application.vue'
import router from './router';


const store = createStore({
    state: {
        headers: {},
        authorizationUser: {},
    },
    mutations: {
        addHeader (state, {name, value}) {
            state.headers[name] = value
        },
        removeHeader (state, {name}) {
            state.headers[name] = null;
        },
```

```
        setAuthorizationUser (state, value) {
            state.authorizationUser = value;
        }
    },
    getters: {
        isAuthenticated (state) {
            const token = state.headers['Authorization'];
            return token !== null && token !== undefined;
        },

        getAuthorization (state) {
            return state.headers['Authorization'];
        },

        getAuthorizationUser (state) {
           return state.authorizationUser;
        }
    }

})

createApp(WebLearnApplication)
    .use(router)
    .use(store)
    .mount('#app')
```

Лістинг файлу router.js

```
import { createWebHistory, createRouter } from 'vue-router'

import HomePage from '@/views/HomePage.vue';
const routes = [
    {
        path: '/',
        component: HomePage
    },
    {
        path: '/login',
        component: () => import('./views/LoginPage'),
    },
    {
        path: '/registration',
        component: () => import('./views/RegistrationPage'),
    },
    {
        path: '/course/create',
        component: () => import('./views/AddCoursesPage'),
    },
    {
        path: '/course/:id',
        component: () => import('./views/ViewCoursePage'),
    },
    {
        path: '/course/:id/edit',
        component: () => import('./views/EditCoursePage'),
    },
    {
        path: '/course/search',
        component: () => import('./components/course/SearchPage'),
```

```
    },
    {
        path: '/user',
        component: () => import('./views/UserPage.vue'),
    },
]

const router = createRouter({
    history:  createWebHistory(),
    routes: routes
});

export default router;
```

Лістинг файлу ViewCoursePage.vue

```
<script>
import axios from 'axios';
import ViewCourseBanner from '@/components/course/ViewCourseBanner.vue'
import RelatedCourseList from "@/components/course/RelatedCourses.vue";
import ViewCourseChapter from "@/components/course/ViewCourseChapter.vue";

export default {
  name: "ViewCoursePage.vue",
  components: {
    ViewCourseChapter,
    RelatedCourseList,
    ViewCourseBanner,
  },
  data() {
    return {
      courses: [],
      course: {}
    }
  },
  mounted() {
    const id = this.$route.params.id;
    axios
        .get('http://localhost:3000/courses/' + id)
        .then(res => {
          console.log('res: ', res)
          this.course = res.data
        })
        .catch(err => console.log(`Error fetching course width id: ${id} from the
server: `, err))

    axios
        .get('http://localhost:3000/courses')
        .then(res => {
          // console.log('res: ', res);
          this.courses = Array.from(res.data).slice(0, 4)
        })
        .catch(error => {
          console.log('Error fetching courses from the server: ', error)
        })
  },
  methods: {
  }
}
</script>
```

```
<template>
  <main class="flex-grow-1 add-courses-page">
    <div class="mt-sm-4">
      <div class="container">
        <ViewCourseBanner v-bind:course="course"/>
        <ViewCourseChapter v-for="chapter of course.chapters" :key="chapter.id"
:chapter="chapter"/>
        <div class="mb-5 mt-5 pb-5">
          <div class="container-title">
            <h2>З Цими курсами переглядають</h2>
          </div>
          <RelatedCourseList v-bind:courses="courses"/>
        </div>
      </div>
    </div>
  </main>
</template>
```

Лістинг файлу UserPage.vue

```
<template>
  <div class="container-user">
    <div class="container-authorized" v-if="this.isLoggedIn()">
      <div class="row-user-parameters">
        <ul class="user-list">
          <li class="email">Email: {{this.getAuthorizationUser().email}}</li>
          <li class="name">Name: {{this.getAuthorizationUser().name}}</li>
          <li class="name_roles">Roles: {{this.getAuthorizationUser().roles}}</li>
        </ul>
      </div>
      <div class="container-courses-list">
        <h2>My Courses</h2>
        <p>{{this.getAuthorizationUser().enrolledCourses}}</p>
      </div>
      <div class="container-courses-enrolled">
        <h2>Enrolled Courses</h2>
      </div>
      <div class="row-button-edit-save">
      </div>
    </div>

    <div class="container-non-authorized" v-else>
      <h1>User is not Authorized!</h1>
    </div>
  </div>
</template>

<script>
import {isLoggedIn, getAuthorizationUser} from "@/authorization.js";


export default {
  name: "UserPage",

  methods: {
    isLoggedIn,
    getAuthorizationUser
```

```
  }
}
</script>
```

Лістинг файлу RegistrationPage.vue

```
<template>
  <main class="flex-grow-1">
    <div class="container mt-sm-5 mb-5">
      <div class="text-center"><span class="display-6 bi bi-person-plus"></span>
        <h1 class="mb-4">Реєстрація</h1></div>
      <div class="row justify-content-center">
        <div class="col-sm-8 col-md-7 col-lg-5">
          <div class="card shadow-sm border-0">
            <div class="card-body m-4">
              <form id="new_user_sign_up_form" class="simple_form
new_user_sign_up_form"
                    @submit.prevent="onSubmit"
                    :novalidate="true" accept-charset="UTF-8" method="post"><input
                  type="hidden" name="authenticity_token"

value="iImVUUR3Y7OalKh75y5q8Axq2oEEW56jXn54LkYlPSJmN8gkUqdKvwwML3HZ8aI7UGMZslKw9s-
NGffdfQgk_Q"
                  autocomplete="off">
                <div class="mb-3 string optional
user_sign_up_form_first_name"><label
                    class="form-label string optional"
                    for="user_sign_up_form_first_name">Введіть Ім'я</label>
                  <input v-model="name"
                    class="form-control shadow-sm string optional" type="text"
                    name="user_sign_up_form[first_name]"
id="user_sign_up_form_first_name"/>
                </div>
                <div class="mb-3 email optional user_sign_up_form_email"><label
                    class="form-label email optional"
for="user_sign_up_form_email">Email</label>
                  <input v-model="email"
                    class="form-control shadow-sm string email optional"
autocomplete="email"
                    type="email" name="user_sign_up_form[email]"
id="user_sign_up_form_email"/>
                </div>
                <div class="mb-3 password required
user_sign_up_form_password"><label
                    class="form-label password required"
for="user_sign_up_form_password">Пароль<abbr
                    title="required"></abbr></label>
                  <input v-model="password"
                    class="form-control shadow-sm password required"
autocomplete="new-password"
                    type="password" name="user_sign_up_form[password]"
id="user_sign_up_form_password"/>
                </div>
                <div class="text-end text-muted small my-4">
                  <span class="me-2">У Вас вже є аккаунт?</span>
                  <router-link class="text-decoration-none"
to="/login">Увійти</router-link>
                </div>
                <div  class="form-group">
                  <input type="submit" name="commit" value="Створити аккаунт"
```

```
                        class="btn btn-primary w-100 mb-1"
                        data-disable-with="Створити аккаунт"/>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </main>
</template>
<script>

import axios from "axios";

export default {
  name: 'RegistrationPage',
  data() {
    return {
      name: '',
      email: '',
      password: ''
    }
  },
  methods: {
    onSubmit() {
      if (this.name !== "" && this.email !== "" && this.password !== "") {
        console.log(`Form submit: name=${this.name}, email=${this.email},
password=${this.password}`);
        axios.post('http://localhost:3000/auth/register', {
          name: this.name,
          email: this.email,
          password: this.password
        }).then(res => {
          console.log('response.data: ', res.data);
          this.$router.push('/login');
        }).catch(error => {
          console.log('Error fetching register: ', error);
        })
      } else {
        console.log("Missing attributes!")
      }
    }
  }
}
</script>
```

Лістинг файлу LoginPage.vue

```
<template>
  <main class="flex-grow-1">
    <div class="container mt-sm-5 mb-5">
      <div class="text-center"><span class="display-6 bi bi-box-arrow-in-
right"></span>
        <h1 class="mb-4 text-center">Вхід</h1></div>
      <div class="row justify-content-center">
        <div class="col-sm-8 col-md-7 col-lg-5">
          <div class="card shadow-sm border-0">
            <div class="card-body m-4">
              <form @submit.prevent="onSubmit" method="post"
```

```
                    class="simple_form new_sign_in_form" id="new_sign_in_form"
                    :novalidate="true" accept-charset="UTF-8">
                <input type="hidden" name="authenticity_token"
                       value="8-LjcbkuiQ4-
6caPI7C_pZAyRThKGKPuwStsqw7xrkEdXL4Er_6gAqhxQYUdb3duzDuGCxzzy4ISTONYNdy3ng"
                       autocomplete="off">
                <div class="mb-3 email required sign_in_form_email">
                  <label class="form-label email required"
for="sign_in_form_email">Email<abbr
                      title="required"></abbr></label>
                  <input v-model="email"
                         class="form-control shadow-sm string email required"
data-testid="email"
                         autocomplete="email" type="email"
name="sign_in_form[email]"
                         id="sign_in_form_email"/>
                </div>
                <div class="mb-3 password required sign_in_form_password"><label
                    class="form-label password required"
for="sign_in_form_password">Пароль<abbr
                    title="required"></abbr></label>
                  <input v-model="password"
                         class="form-control shadow-sm password required" data-
testid="password"
                         autocomplete="current-password" type="password"
name="sign_in_form[password]"
                         id="sign_in_form_password"/>
                </div>
                <div class="form-group">
                  <input
                      type="submit" name="commit" value="Увійти" class="btn btn-
primary w-100"
                      data-testid="submit" data-disable-with="Увійти">
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </main>
</template>
<script>
import axios from 'axios';
import {fetchAuthorizationUser} from "@/authorization.js";

export default {
  name: 'LoginPage',
  data() {
    return {
      email: '',
      password: ''
    }
  },
  methods: {
    onSubmit() {
      if (this.email !== '' && this.password !== '') {
        axios.post('http://localhost:3000/auth/login', {
          email: this.email,
          password: this.password
        }).then(res => {
          const token = res.data.token;
          this.$store.commit('addHeader', {name: 'Authorization', value: 'Bearer '
+ token});
```

```
          this.fetchAuthorizationUser();
          this.$router.push('/');
        }).catch(error => {
          console.log('Error fetching login: ', error);
        })
      }
    },
    fetchAuthorizationUser
  }
}
</script>
```

Лістинг файлу HomePage.vue

```
<template>
  <main class="flex-grow-1">
    <div class="mt-sm-5 mb-5">
      <div class="container banner">
        <div class="banner-wrap align-items-center border shadow bg-white rounded-
3">
          <div class="row">
            <div class="col-12 p-5"><h1 class="h6 text-muted text-
center">Безкоштовні Курси Web
              Програмування для початківців</h1>
              <div class="display-4 fw-bold lh-1 mb-3 justify-content-center text-
center">Вивчай  <span class="text-primary"><span
                  class="hero-header">JS</span><span class="typed-cursor"
                                                     aria-
hidden="true">|</span></span></div>
              <p class="lead text-center">Вивчення основ веб-програмування для
новачків. Використання онлайн-курсів з
                можливістю виконання вправ безпосередньо у веб-браузері. Практичні
завдання доступні після
                кожного розділу для закріплення отриманої інформації.</p>
              <div class="d-grid gap-2 d-md-flex justify-content-center"><a
                  class="btn btn-primary btn-lg px-4 me-md-2 fw-bold"
href="#courses">Почати зараз!</a>
              </div>
            </div>
            <!--                    <div class="col-lg-4 offset-lg-1 p-0 overflow-
hidden shadow-lg text-center"><img-->
            <!--                                alt="" class="rounded-lg-3"
height="356"-->
            <!--                                src="https://code-
basics.com/assets/code-basics-coding-ru-
656fd568b9a35ff40391739b52842ba90093158053168d3269fe2cd6600826c6.png"-->
            <!--                                width="720">-->
          </div>
        </div>
      </div>
    </div>
    <div class="mb-5 pb-5">
      <CourseList v-bind:courses="courses" />
    </div>
  </main>
</template>
<script>

import axios from 'axios';
import CourseList from '@/components/course/CourseList.vue';
```

```
export default {
  name: 'HomePage',
  data() {
    return {
      courses: []
    }
  },
  mounted() {
    axios
        .get('http://localhost:3000/courses')
        .then(res => {
          // console.log('res: ', res);
          this.courses = res.data
        })
        .catch(error => {
          console.log('Error fetching courses from the server: ', error)
        })
  },
  components: {
    CourseList
  }
}
</script>
```

Лістинг файлу EditCoursePage.vue

```
<script>
import EditCourseChapter from "@/views/EditCourseChapter.vue";
import EditCourseBanner from "@/components/course/EditCourseBanner.vue";
import axios from "axios";

export default {
  name: 'EditCoursePage',
  data() {
    return {
      course: {},
    }
  },
  mounted() {
    const id = this.$route.params.id;
    axios
        .get('http://localhost:3000/courses/' + id)
        .then(res => {
          console.log('res: ', res)
          this.course = res.data
          this.selectLastParagraph(this.course);
          console.log(this.course);
        })
        .catch(err => console.log(`Error fetching course width id: ${id} from the
server: `, err))
  },
  components: {
    EditCourseChapter,
    EditCourseBanner
  },

  methods: {
    addChapter() {
      this.course.chapters.push(this.createChapter());
```

```
    },
    removeChapter(index) {
      this.course.chapters.splice(index, 1);
    },

    selectLastParagraph (course) {
      course.chapters.forEach(chapter => {
        chapter.paragraphs.forEach((paragraph, index) => {
          if (index === chapter.paragraphs.length - 1) {
            paragraph.selected = true;
          }
          else  {
           paragraph.selected = false;
          }
        });
      });
    },

    createChapter() {
      return {
        title: '',
        paragraphs: [
          {
            title: '',
            text: '',
            selected: true,
          }
        ]
      };
    },

    saveCourse() {
      //TODO add validation...
      const body = JSON.parse(JSON.stringify(this.course));
      body.chapters.forEach(chapter => {
        chapter.paragraphs.forEach(paragraph => {
          delete paragraph.selected;
        });
      });

      console.log("Updating course: ", body);
      axios
          .put(`http://localhost:3000/courses/${this.course.id}`, body)
          .then(res => {
            if (res.status === 200) {
              const data = res.data;
              console.log("Successfully updated course: ", data);
              this.$router.push('/course/' + data.id);
            }
          })
          .catch(err => console.log("Error updating course: ", err));
    }
  }
}


</script>

<template>
<div><h1 class="d-flex text-center m-1-auto justify-content-center title-edit-
course">Редагування Курсу</h1></div>
  <main class="flex-grow-1 add-courses-page">
    <div class="mt-sm-4">
      <div class="container">
```

```
        <EditCourseBanner :course-image-url="course.courseImageUrl"
                          :save-course="this.saveCourse"
                          @change-course-image-url="(url) => course.courseImageUrl
= url"
                          @set-course-name="(name) => course.name = name"
                          @set-course-description="(description) =>
course.description = description"
                          @set-course-duration="(duration) => course.duration =
duration"
                          :save-course-text="'Редагувати Курс'"
                          :course-name="`${course.name}`"
                          :course-description="`${course.description}`"
                          :course-duration="`${course.duration}`"

        />
        <EditCourseChapter v-for="(chapter, index) of this.course.chapters"
:key="chapter.id"
                           :chapter="chapter" :index="index"
                           :can-be-deleted="this.course.chapters.length > 1"
@remove-chapter="(i) => this.removeChapter(i)"

        />
        <div class="row d-flex justify-content-center mb-3">
          <button class="col-auto btn btn-outline-primary"
@click.prevent="addChapter">Додати розділ</button>
        </div>
      </div>
    </div>
  </main>
</template>

<style scoped>

</style>
```

Лістинг файлу EditCourseChapter.vue

```
<template>
  <div>
    <div class="row mt-5 col-input-info">
      <div class="col-12 d-flex justify-content-between align-items-center">
        <input v-model="this.$props.chapter.title" id="chapterTitle"
placeholder="Введіть назву розділа"/>
        <img v-if="this.$props.canBeDeleted" @click="this.$emit('removeChapter',
this.$props.index)" src="/img/trash_icon.png" width="32" height="32" class="ms-2
img-thumbnail" alt="trash">
      </div>
    </div>
    <div class="row mb-5">
      <div class="col-3 mt-4">
        <ul class="list-group">
          <EditChapterParagraph v-for="(paragraph, index) of
this.$props.chapter.paragraphs" :key="index"
                                :index="index"
                                :paragraph="paragraph"
                                :chapter-
paragraphs="this.$props.chapter.paragraphs"
                                :select-paragraph="selectParagraph"
          />
          <li class="list-group-item d-flex justify-content-center"
```

```
@click="addParagraph">
            <img src="/img/plus_icon.png" class="img-thumbnail" width="28"
height="28">
          </li>
        </ul>
      </div>
      <div class="col col-input-info mt-4">
        <TextEditor v-bind:selectedParagraphContent="selectedParagraphContent"/>
      </div>
    </div>
  </div>
</template>
<script>
import EditChapterParagraph from "@/components/course/EditChapterParagraph.vue"
import TextEditor from '@/components/parts/TextEditor.vue'

export default {
  name: 'EditCourseChapter',
  components: {EditChapterParagraph, TextEditor},
  computed: {
    selectedParagraphContent () {
      const paragraphs = this.$props.chapter.paragraphs;
      let selectedIndex = paragraphs.findIndex((paragraph) => paragraph.selected);
      return paragraphs[selectedIndex];
    }
  },
  props: {
    chapter: {
      type: Object,
      required: true
    },
    index: {
      type: Number,
      required: true
    },
    canBeDeleted: {
      type: Boolean,
      required: false
    }
  },
  methods: {
    deselectChapterParagraphs() {
      this.$props.chapter.paragraphs.forEach(paragraph => {
        paragraph.selected = false
      })
    },
    addParagraph: function () {
      this.deselectChapterParagraphs();

      this.$props.chapter.paragraphs.push({
        title: '',
        text: '',
        selected: true,
      });
    },
    selectParagraph: function (index) {
      this.deselectChapterParagraphs()
      this.$props.chapter.paragraphs[index].selected = true;
    },
  }
}
</script>
<style scoped>
```

```
.list-group li {
  cursor: pointer;
}

</style>
```

Лістинг файлу AddCoursePage.vue

```
<template>
  <main class="flex-grow-1 add-courses-page">
    <div class="mt-sm-4">
      <div class="container">
        <EditCourseBanner :course-image-url="course.courseImageUrl"
                          :save-course="this.addCourse"
                          @change-course-image-url="(url) => course.courseImageUrl
= url"
                          @set-course-name="(name) => course.name = name"
                          @set-course-description="(description) =>
course.description = description"
                          @set-course-duration="(duration) => course.duration =
duration"
                          :save-course-text="'Додати Курс'"/>
        <EditCourseChapter v-for="(chapter, index) of this.course.chapters"
:key="chapter.id"
                           :chapter="chapter" :index="index"
                           :can-be-deleted="this.course.chapters.length > 1"
@remove-chapter="(i) => this.removeChapter(i)"

        />
        <div class="row d-flex justify-content-center mb-3">
          <button class="col-auto btn btn-outline-primary"
@click.prevent="addChapter">Додати розділ</button>
        </div>
      </div>
    </div>
  </main>
</template>

<script>

// eslint-disable-next-line no-unused-vars
import axios from 'axios';

import EditCourseBanner from "@/components/course/EditCourseBanner.vue";
import EditCourseChapter from "@/views/EditCourseChapter.vue";

export default {
  name: 'AddCoursesPage',
  data() {
    return {
      course: {
        name: '',
        description: '',
        courseImageUrl: '/img/add_image.jpg',
        duration: '',
        chapters: [
          this.createChapter(),
        ]
      },
    }
  },
  components: {
    EditCourseChapter,
```

```
    EditCourseBanner
  },

  methods: {
    addChapter() {
      this.course.chapters.push(this.createChapter());
    },
    removeChapter(index) {
      this.course.chapters.splice(index, 1);
    },
    createChapter() {
      return {
        title: '',
        paragraphs: [
          {
            title: '',
            text: '',
            selected: true,
          }
        ]
      };
    },
    addCourse() {
      //TODO add validation...
      const body = JSON.parse(JSON.stringify(this.course));
      body.chapters.forEach(chapter => {
        chapter.paragraphs.forEach(paragraph => {
          delete paragraph.selected;
        });
      });

      console.log("Adding course: ", body);
      axios
          .post('http://localhost:3000/courses', body)
          .then(res => {
            if (res.status === 201) {
              const data = res.data;
              console.log("Successfully added course: ", data);
              this.$router.push('/courses/' + data.id);
            }
          })
          .catch(err => console.log("Error adding course: ", err));
    }
  }
}
</script>

<style scoped>

.list-group li {
  cursor: pointer;
}

</style>
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА НАВЧАЛЬНОЇ СИСТЕМИ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБПРОГРАМУВАННЯ

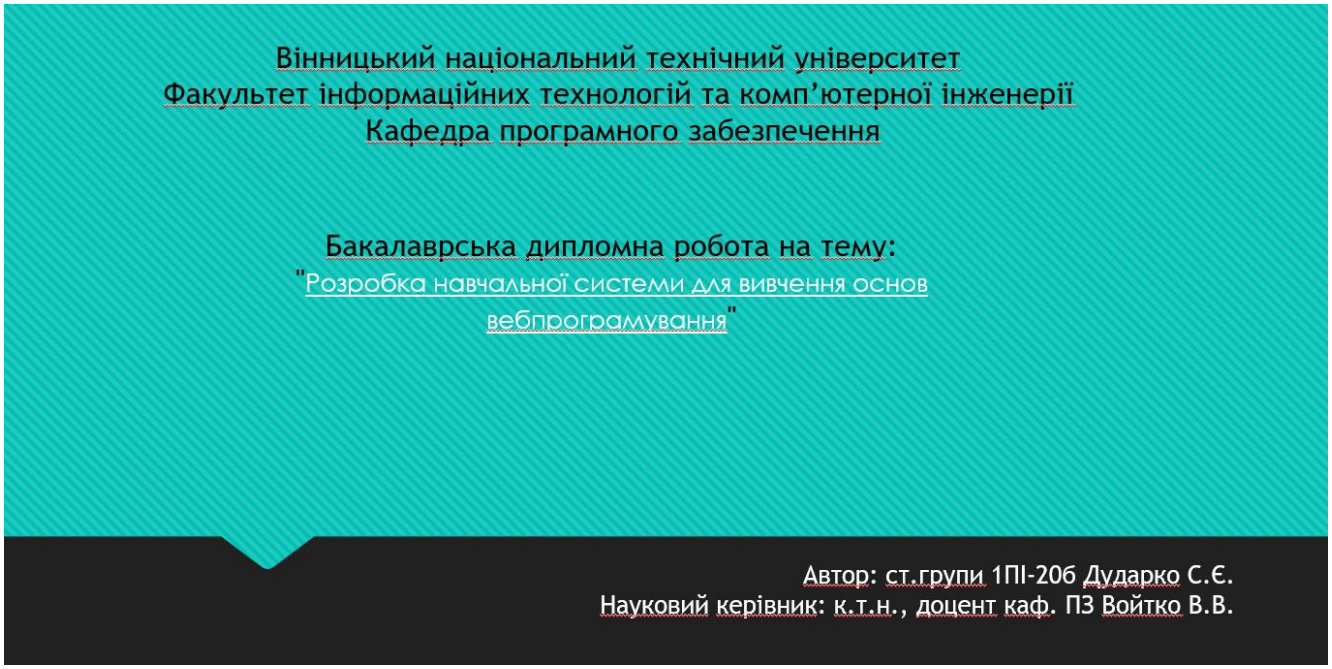


Рисунок Г.1 – Титульний слайд

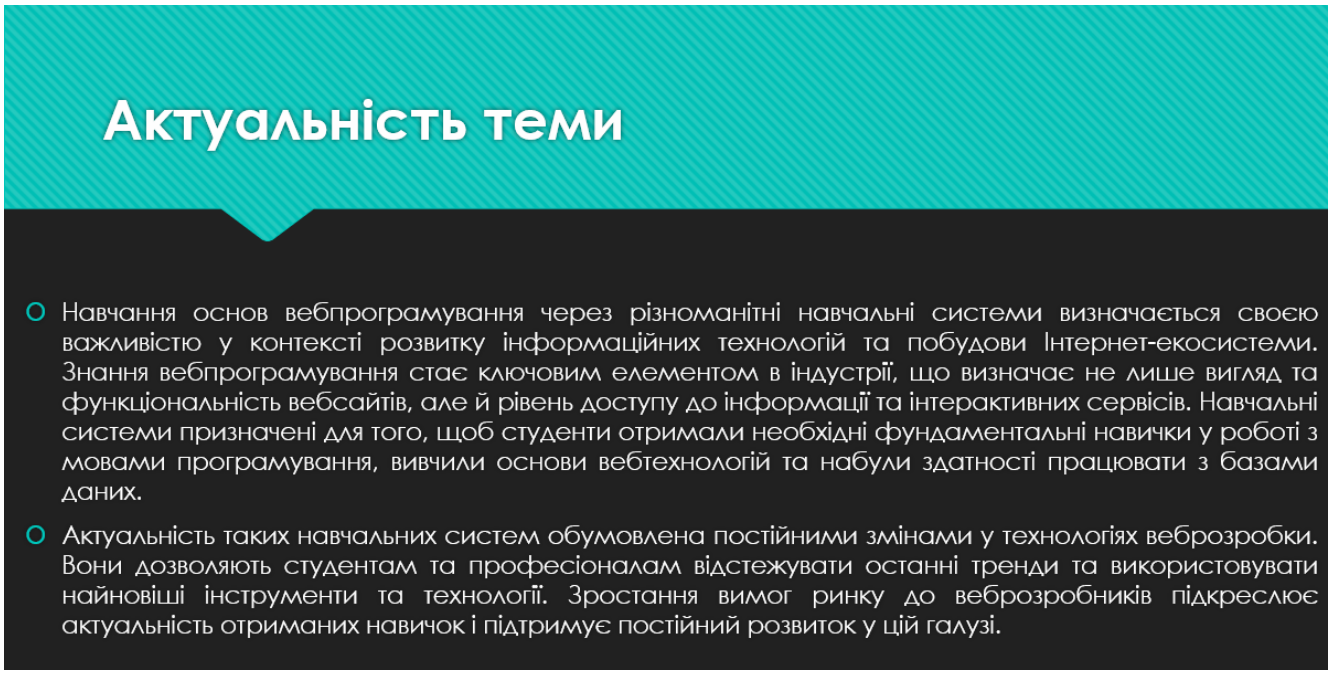


Рисунок Г.2 – Актуальність теми

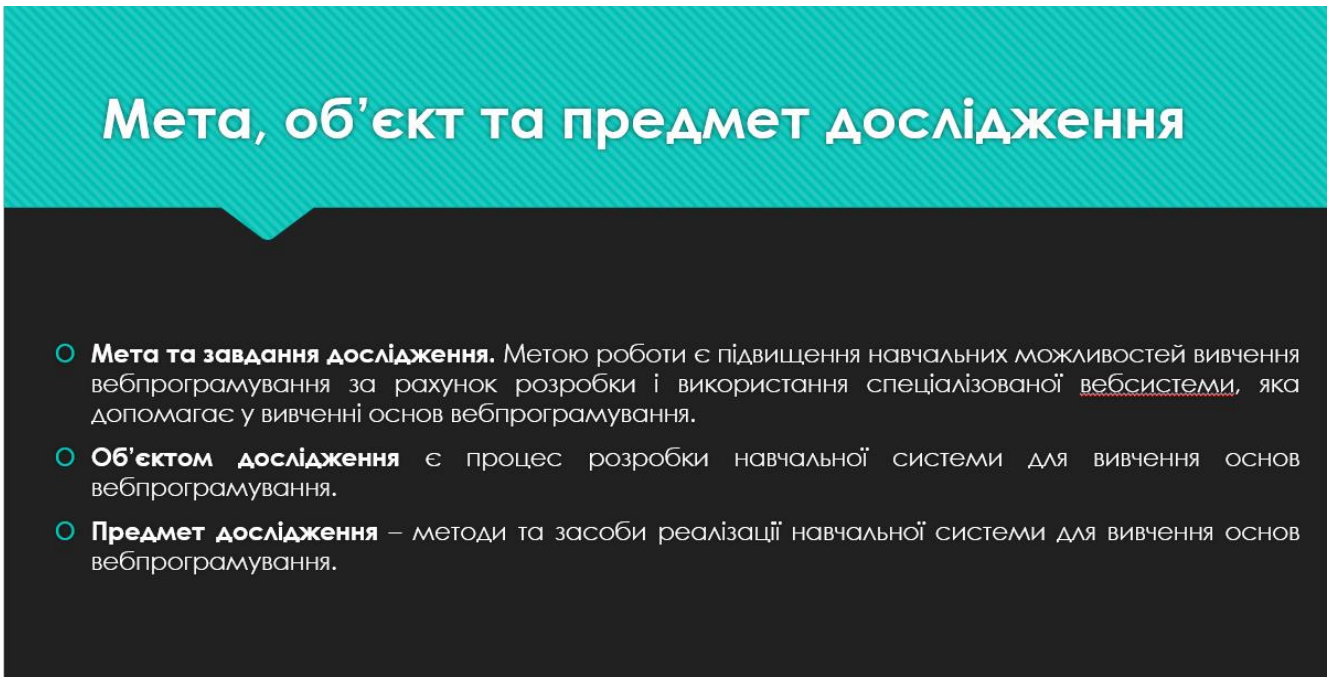


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

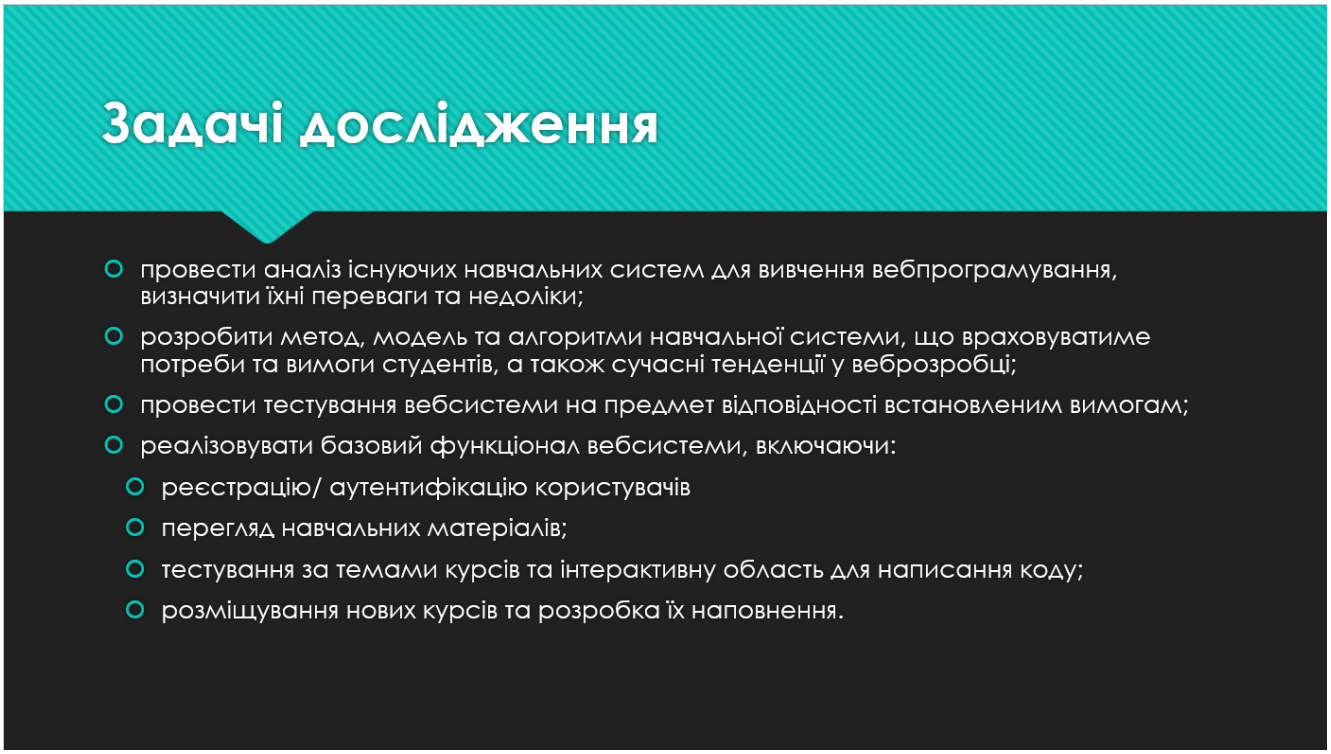


Рисунок Г.4 – Задачі дослідження

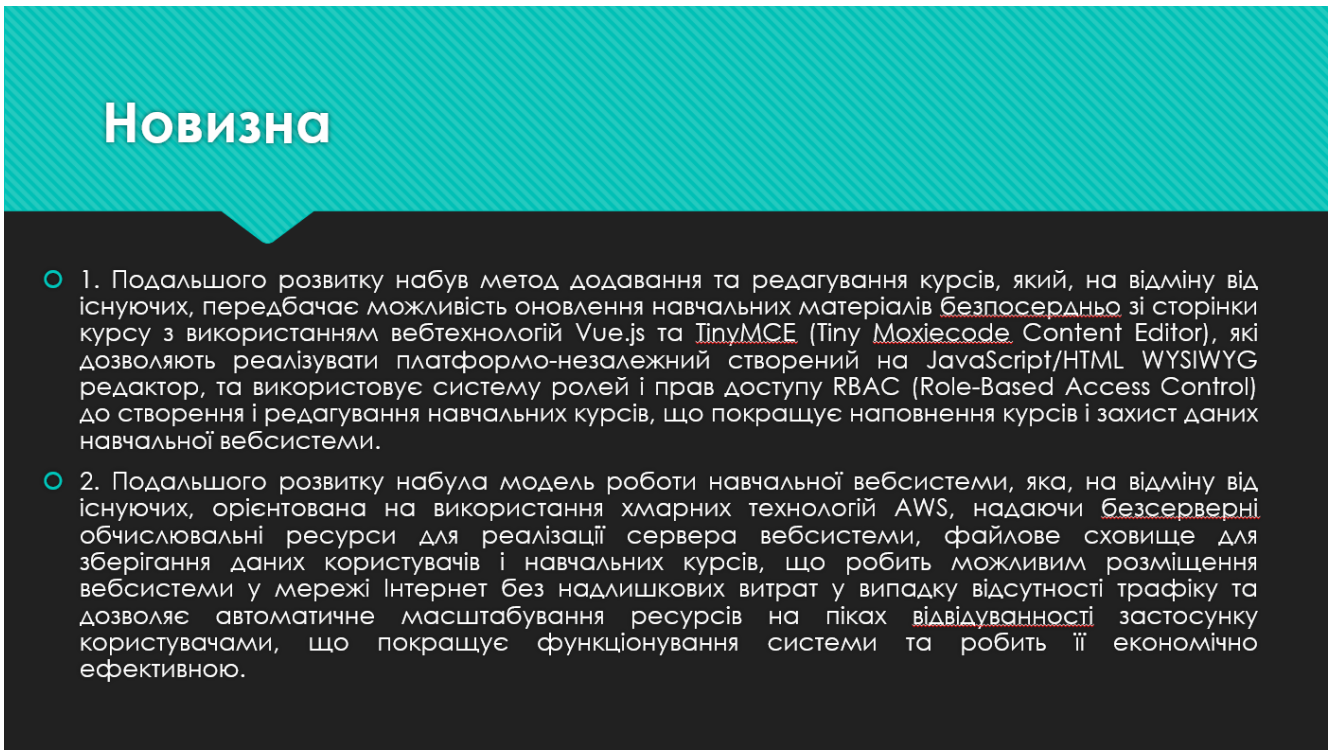


Рисунок Г.5 – Новизна

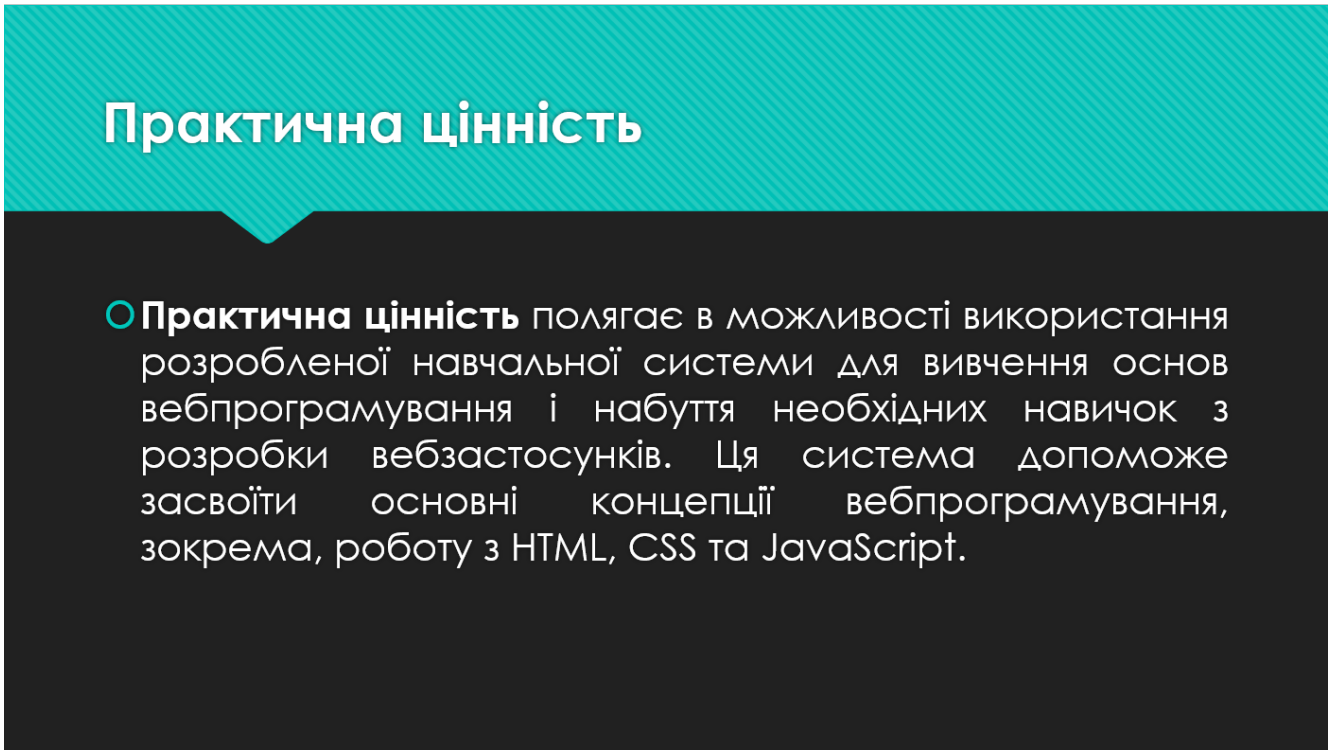


Рисунок Г.6 – Практична цінність

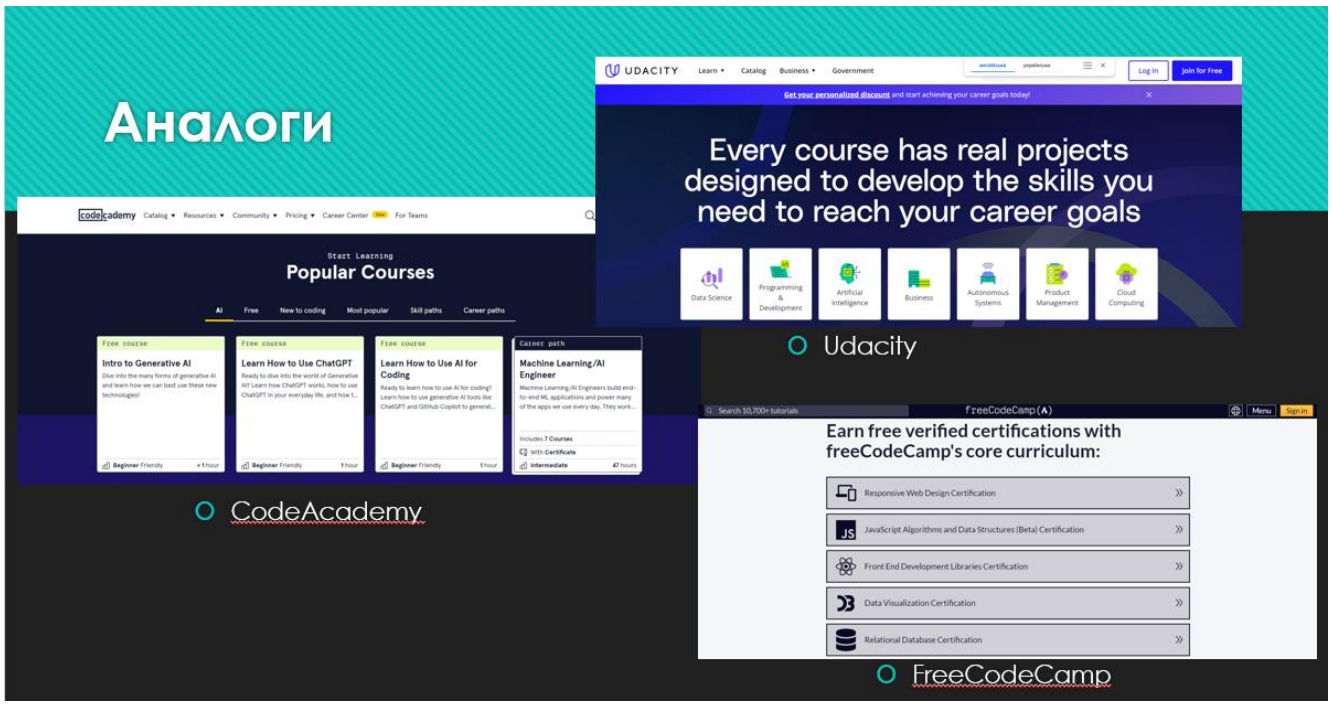


Рисунок Г.7 – Аналоги

Критерій	Codeacademy	Udacity	freeCodeCamp	Власна розробка
1	2	3	4	5
Безкоштовний доступ	0	0	1	1
Спільнота користувачів	1	0	1	1
Проходження практичних завдань користувачами	0	1	0	1
Проходження питань для самоконтролю користувачами	1	1	1	1
Відслідковування навчального прогресу та результатів тестування	1	1	1	1
Сумарний коефіцієнт	3	3	4	5

Рисунок Г.8 – Порівняльний аналіз аналогів

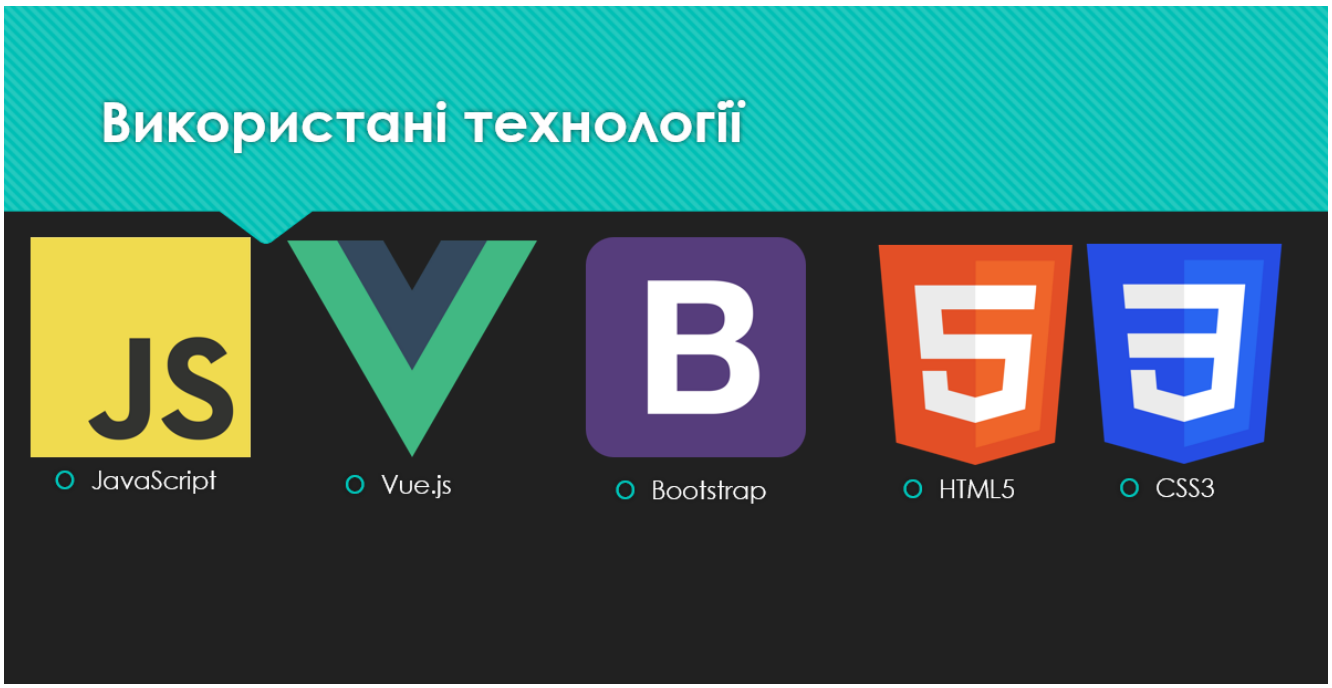


Рисунок Г.9 – Використані технології

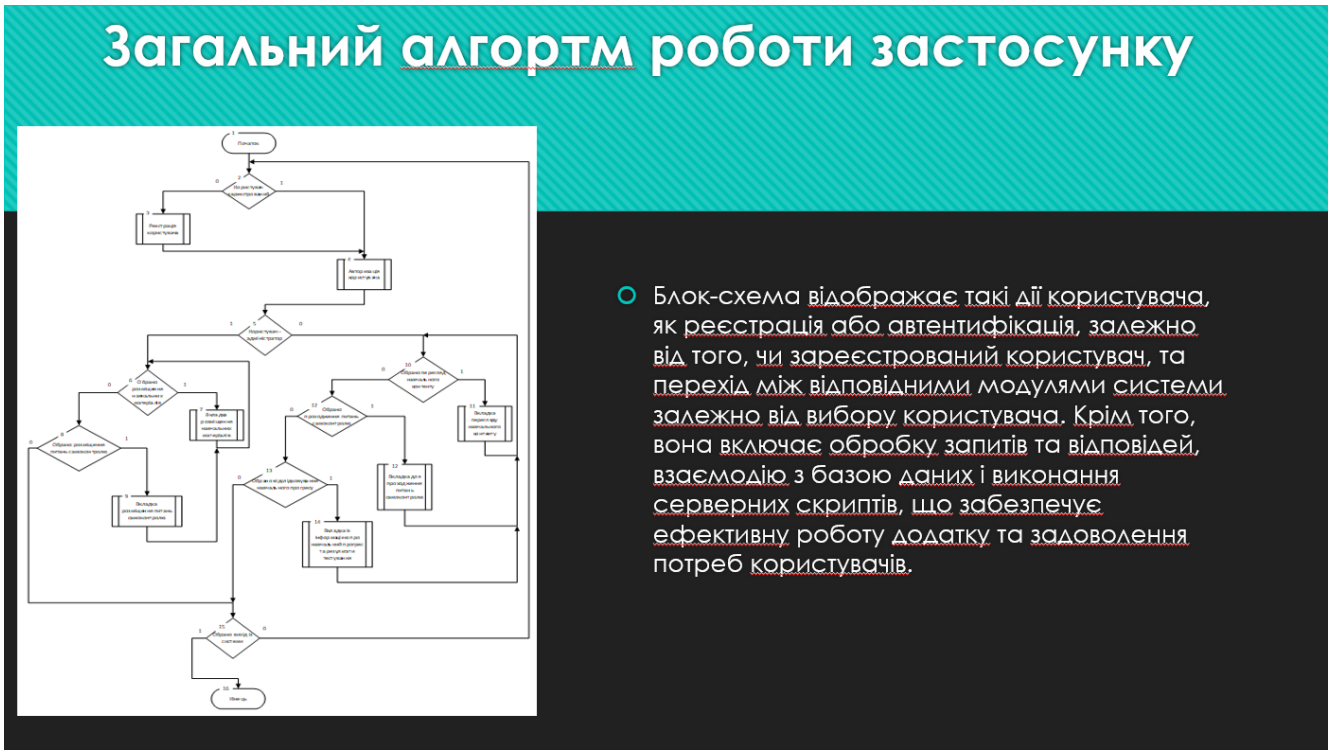


Рисунок Г.10 – Загальний алгортм роботи застосунку

Алгоритм процесу реєстрації користувача

- Реєстрація є першим етапом, який проходить користувач перед використанням системи. Він має можливість зареєструватися, використовуючи логін та пароль. Процес реєстрації складається з кількох кроків:
- 1. Введення персональних даних: користувач заповнює поля форми реєстрації, де вказує своє ім'я, електронну пошту, логін та пароль. Можливо, система потребує підтвердження пароля, щоб уникнути помилок під час введення.
- 2. Вибір логіну та пароля: користувач вибирає унікальний логін, який буде використовуватися для входу в систему, та створює надійний пароль. Пароль має відповідати певним вимогам безпеки, таким як мінімальна довжина, наявність цифр та спеціальних символів.

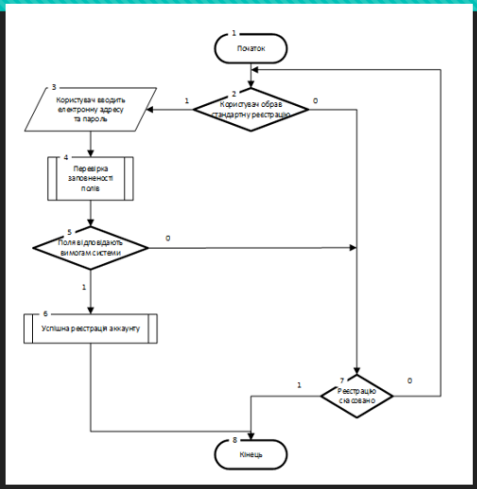
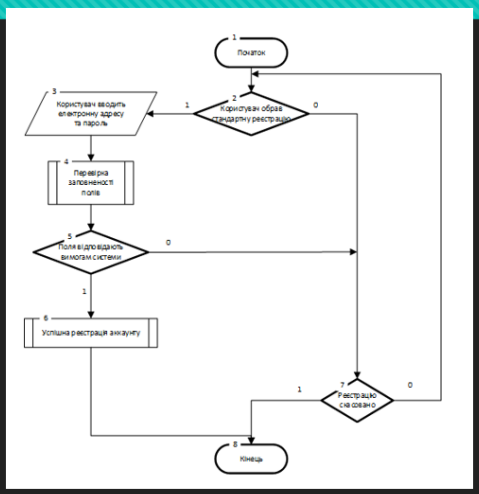


Рисунок Г.11 – Алгоритм процесу реєстрації користувача

Алгоритм процесу авторизації користувача



- **Після успішної реєстрації користувача наступним кроком є авторизація. Це процес перевірки автентичності, де користувач надає свої облікові дані (Email та пароль), щоб отримати доступ до системи або ресурсу.**

Рисунок Г.12 – Алгоритм процесу авторизації користувача

Алгоритм проходження навчального тестування

- Компонент TestForm.vue є основним і відображає список питань, кожне з яких представлено окремим компонентом Question.vue. Після завершення тестування, результати відображаються за допомогою компоненту Result.vue.
- - FetchQuestions(). Метод викликається для отримання списку питань з сервера. Використовує Axios для відправлення GET запиту до API.
- - SubmitAnswers(). Метод викликається при відправленні відповідей на сервер. Використовує Axios для відправлення POST запиту з відповідями користувача.
- - CalculateScore(). Метод обчислює коефіцієнт результату на основі правильних відповідей. Реалізація цього методу залежить від логіки оцінювання конкретного тесту.
- Використання Axios для відправлення відповідей на сервер (axios.post) та отримання питань (axios.get). API надає endpoints для отримання списку питань та відправлення відповідей.
- Модуль забезпечує гнучке та адаптивне середовище для проведення тестування, дозволяючи користувачам ефективно моніторити свій прогрес та отримувати відгук про рівень своїх знань або навичок.

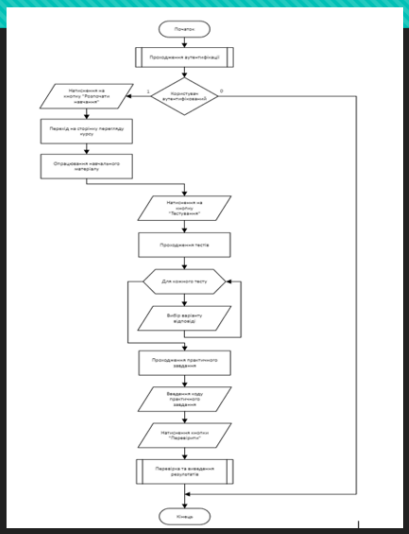
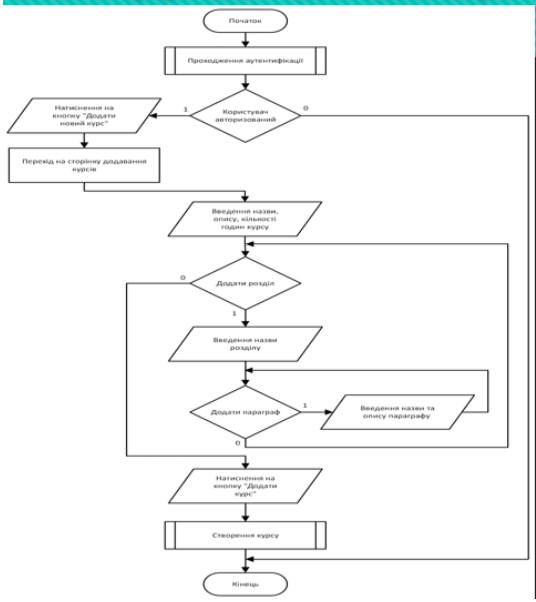


Рисунок Г.13 – Алгоритм проходження навчального тестування

Метод додавання та редагування курсів із авторизацією



- 1. Користувач використовує метод authenticate класу User для перевірки своїх облікових даних (ім'я користувача та пароль) збережених у базі даних (users_db). Якщо дані вірні, користувач отримує доступ до системи.
- 2. Перевірка прав доступу користувача відбувається автоматично під час аутентифікації. Якщо користувач має роль admin, він має право створювати нові курси.
- 3. Користувач натискає на кнопку "Додати новий курс", що спрацьовує метод add_course у головній програмі, який переадресовує користувача на сторінку створення курсу.
- 4. Користувач заповнює поля курсу: назву (title), опис (description) та кількість годин (hours). Ця інформація зберігається у локальних змінних об'єкта course.
- 5. Для кожного розділу викликається метод add_section класу Course, який додає новий розділ з назвою та списком параграфів. Параграфи додаються за допомогою методу add_paragraph, який додає параграф з назвою та описом до поточного розділу.
- 6. Для кожного параграфа викликається метод add_question, який додає тестове питання з варіантами відповідей. Правильний варіант відмічується спеціальним прапорцем.
- 7. Зберігання курсу: Після завершення всіх кроків, користувач натискає на кнопку "Додати курс". Головна програма викликає метод save_courses, який зберігає інформацію про курс у форматі JSON у файлі courses.json через виклик HTTP методу POST /courses.

Рисунок Г.14 – Метод додавання та редагування курсів із авторизацією

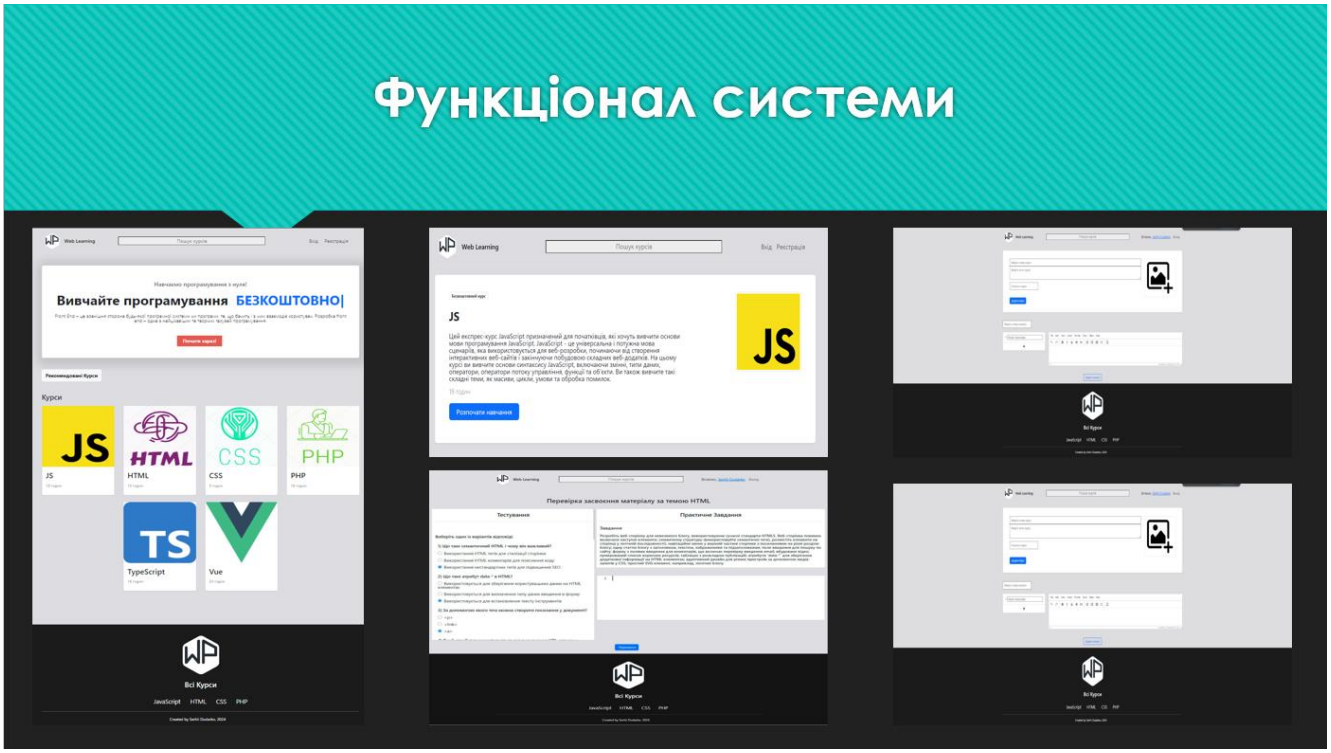


Рисунок Г.15 – Функціонал системи

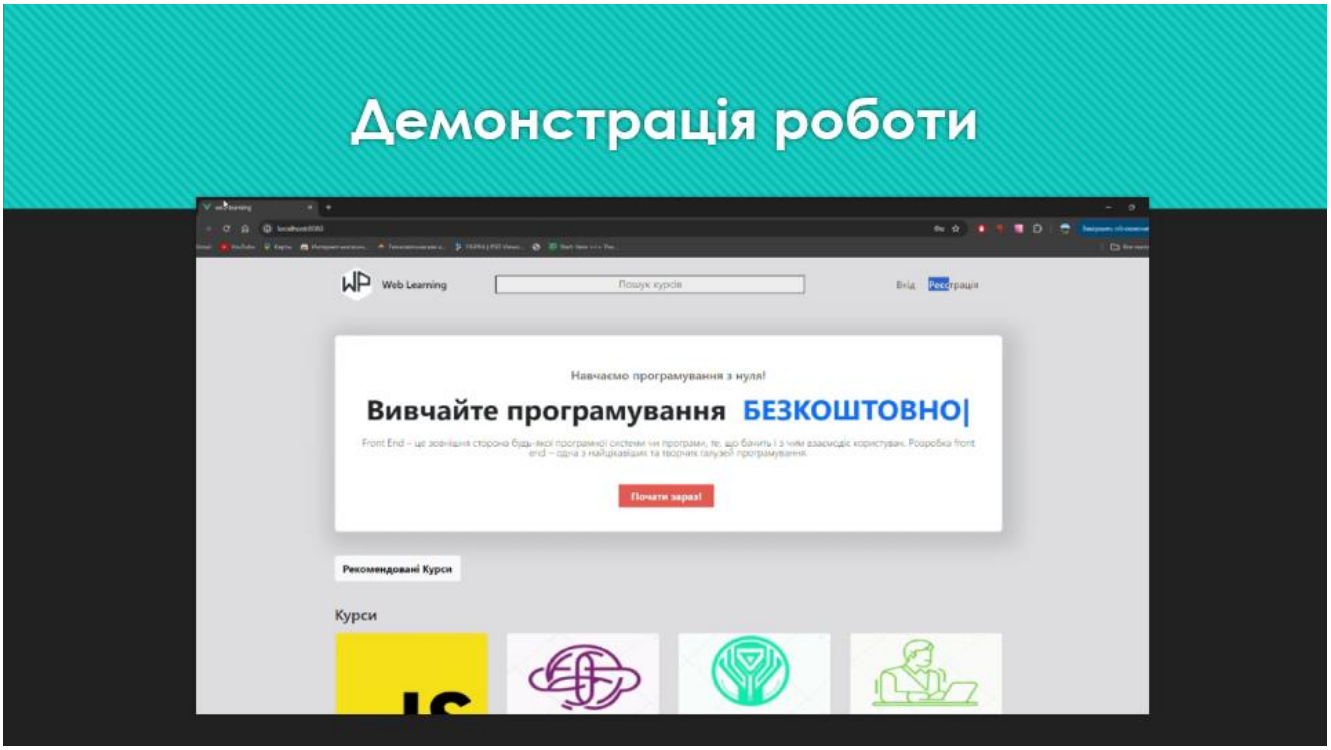


Рисунок Г.16 – Демонстрація роботи

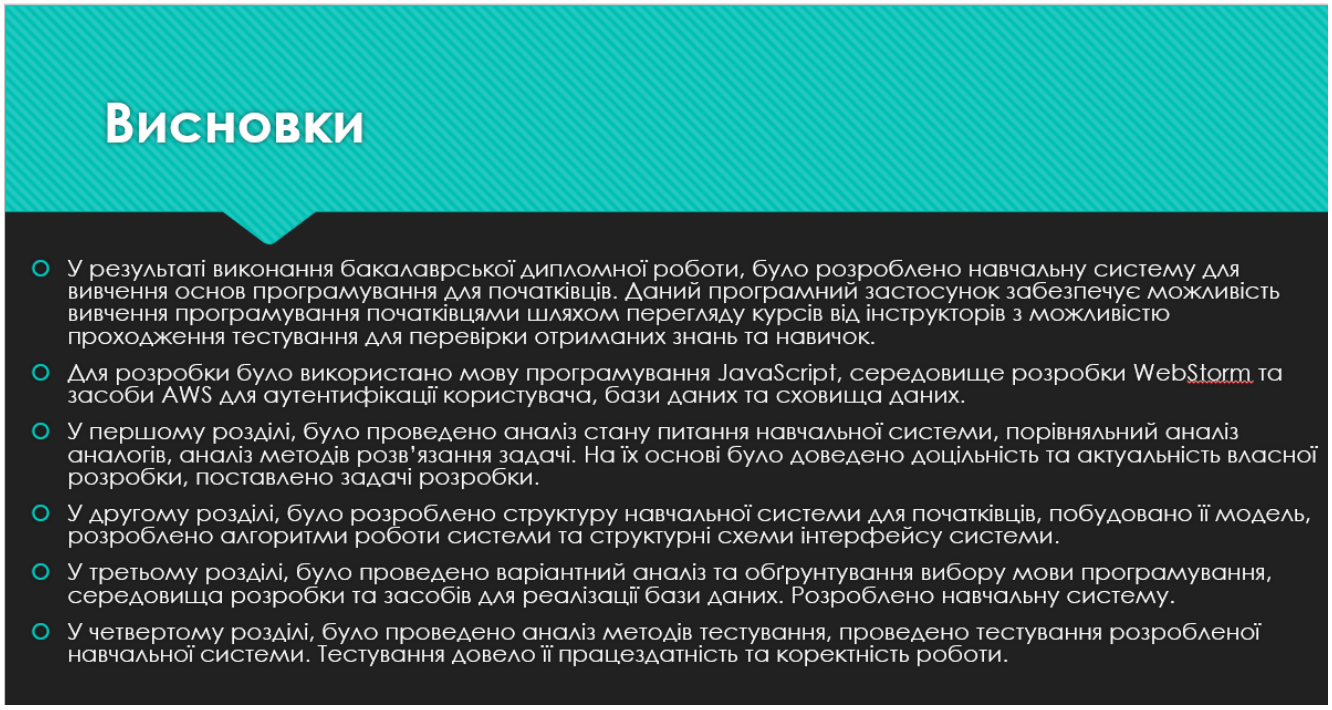


Рисунок Г.17 – Висновки (частина 1)

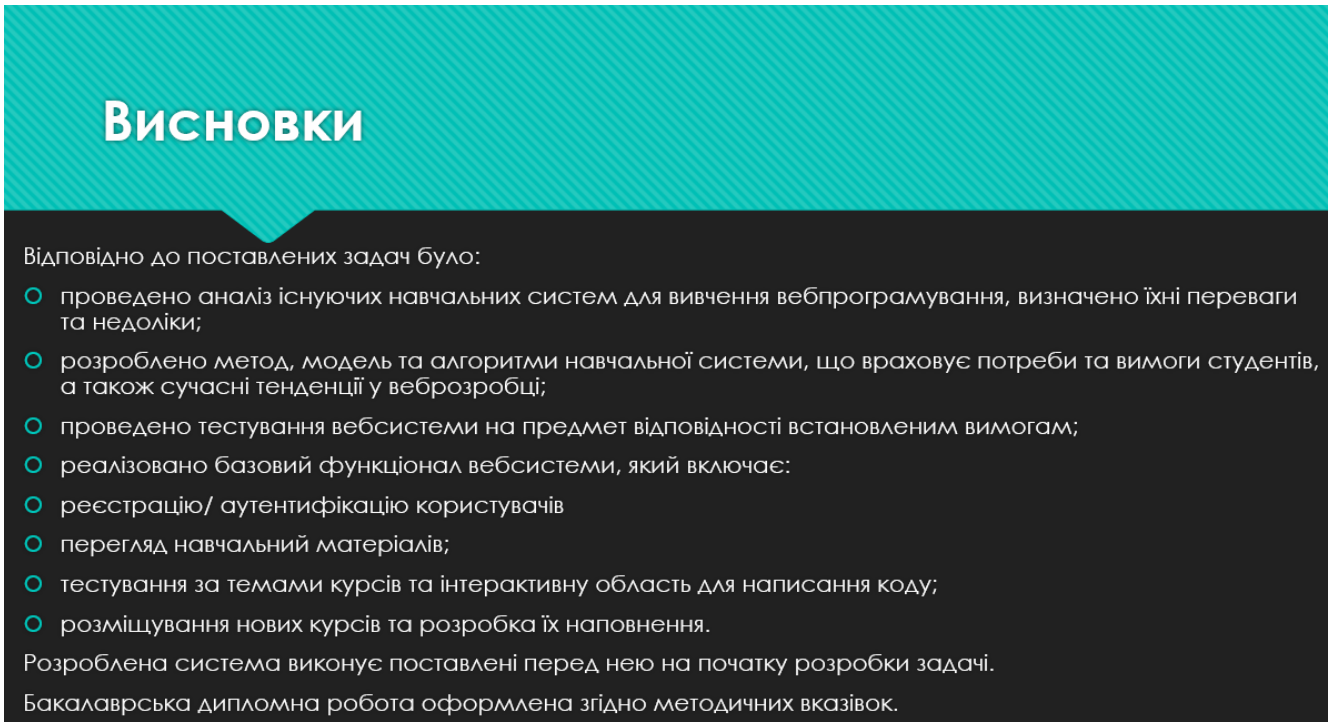


Рисунок Г.18 – Висновки (частина 2)

Апробація результатів роботи і публікації

- **Апробація результатів роботи.** Описані у бакалаврській дипломній роботі положення доповідались на конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)».
- **Публікації**. Результати роботи були опубліковані в матеріалах конференції – LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024).


ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

СЕРТИФІКАТ

засвідчує, що

Сергій ДУДАРКО

взяв участь у роботі

LIII Всеукраїнської науково-технічної конференції підрозділів ВНТУ

20 — 22 березня 2024 р.

Тривалість 15 годин (0,5 кредиту ЄКТС)

Проректор з наукової роботи ВНТУ Ірина ЄПІФАНОВА

Рисунок Г.19 – Апробація результатів роботи і публікації