

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
" _____ " _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА
БАКАЛАВРА

**РОЗРОБКА ДОДАТКУ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБ-
ПРОГРАМУВАННЯ**

Виконав – студент 2 курсу, групи 22к-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Клапоусь Віталій Дмитрович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2025

Зміст

ВСТУП	3
РОЗДІЛ 1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБ-СИСТЕМ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБ-ПРОГРАМУВАННЯ	5
1.1. Аналіз предметної області	5
1.2. Порівняльний аналіз аналогів.....	6
1.3. Постановка задачі	10
РОЗДІЛ 2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ	13
2.1. Аналіз даних та розробка моделі системи.....	13
2.2. Розробка механізму керування навчальними курсами з урахуванням процедур автентифікації та авторизації користувачів	16
2.3. Розробка загальних алгоритмічних рішень для забезпечення функціонування застосунку	18
2.4. Аналіз та обґрунтування вибору мови програмування.....	22
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ	25
3.1. Реалізація модуля автентифікації та реєстрації користувачів.....	25
3.2. Реалізація модуля для формування навчальних курсів	29
3.3. Реалізація модуля для проходження тестування	32
3.4. Тестування додатку для опанування основ веб-програмування.....	34
ВИСНОВКИ.....	41
СПИСОК ЛІТЕРАТУРИ.....	43
ДОДАТКИ.....	46

ВСТУП

У сучасних умовах динамічного розвитку інформаційних технологій особливої ваги набуває питання формування цифрової грамотності населення. Одним із ключових аспектів цієї грамотності є володіння основами веб-програмування, яке забезпечує розуміння принципів функціонування сучасних інформаційних систем, сервісів та ресурсів. Знання HTML, CSS і JavaScript стають не лише перевагою для спеціалістів ІТ-сфери, а й важливим інструментом для представників інших професійних галузей.

З огляду на це, виникає потреба у створенні ефективних засобів для навчання, які відповідатимуть сучасним вимогам доступності, інтерактивності та гнучкості. Особливого значення набувають програмні додатки, здатні поєднувати теоретичний матеріал із практичними завданнями. Актуальність теми визначається також загальною тенденцією до цифровізації освіти, розвитком дистанційних та змішаних форм навчання, а також зростаючим попитом на самостійне опанування нових знань і навичок.

Таким чином, розробка спеціалізованого додатку для вивчення основ веб-програмування є своєчасною відповіддю на запити як освітнього середовища, так і ринку праці.

Метою роботи є створення додатку, призначеного для полегшення процесу навчання основам веб-програмування, що сприятиме підвищенню ефективності засвоєння теоретичних знань і розвитку практичних навичок користувачів.

Об'єктом є процес навчання основ веб-програмування з використанням інформаційних технологій.

Предметом дослідження є програмний додаток як засіб організації навчального процесу з вивчення основ веб-програмування, його структура, функціональність та ефективність застосування.

Для досягнення поставленої мети в роботі використано комплекс наукових методів, а саме: аналіз літературних джерел та електронних ресурсів, присвячених методикам навчання програмуванню, особливостям використання програмних засобів у навчальному процесі, а також технічним аспектам веб-розробки; порівняльний аналіз існуючих програмних продуктів, орієнтованих на навчання веб-програмуванню, з метою визначення їх сильних і слабких сторін; методи системного та структурного аналізу при проєктуванні архітектури додатку та його функціональних модулів; методи програмної інженерії для безпосередньої розробки програмного продукту; методи тестування для перевірки працездатності, надійності та зручності використання розробленого додатку; методи анкетування або опитування (за потреби) для оцінки ефективності додатку з боку потенційних користувачів.

Практичне значення дослідження полягає у можливості використання розробленої навчальної системи як ефективного засобу для формування та розвитку початкових знань і практичних навичок у сфері веб-програмування. Функціонал системи орієнтований на засвоєння базових принципів створення веб-застосунків, зокрема опанування основних елементів і технологій, таких як HTML, CSS та JavaScript.

РОЗДІЛ 1 АНАЛІЗ СТАНУ РОЗВИТКУ ВЕБ-СИСТЕМ ДЛЯ ВИВЧЕННЯ ОСНОВ ВЕБ-ПРОГРАМУВАННЯ

1.1. Аналіз предметної області

У сучасних умовах стрімкого розвитку інформаційних технологій та цифрової освіти значна увага приділяється створенню й удосконаленню веб-систем, призначених для навчання програмуванню. Веб-програмування, як одна з ключових галузей ІТ-сфери, потребує доступних та ефективних засобів навчання, що спрямовані на формування як теоретичних знань, так і практичних навичок. Відповідно, з кожним роком спостерігається зростання кількості освітніх платформ, онлайн-курсів, інтерактивних середовищ та спеціалізованих додатків, орієнтованих саме на вивчення основ веб-розробки.

Стан розвитку веб-систем для навчання основам веб-програмування можна охарактеризувати як активний та динамічний. Наразі існує велика кількість платформ, які пропонують користувачам як теоретичний матеріал, так і практичні вправи. Серед найбільш відомих ресурсів варто виділити такі платформи, як Codecademy, FreeCodeCamp, W3Schools, Sololearn та Khan Academy. Кожна з цих систем має власний підхід до структурування навчального матеріалу, форматів взаємодії з користувачем та механізмів перевірки знань.

Зокрема, Codecademy надає доступ до інтерактивних курсів, які включають поступове ознайомлення з теорією та виконання практичних завдань у вбудованому редакторі коду. Платформа FreeCodeCamp відома своєю великою базою навчальних проєктів, що дає змогу закріпити знання шляхом створення реальних застосунків. W3Schools орієнтована переважно на довідковий матеріал і приклади коду, що робить її корисною для швидкого пошуку інформації, однак менш інтерактивною порівняно з іншими ресурсами. Sololearn позиціонує себе як мобільний навчальний додаток із короткими інтерактивними уроками, що особливо зручно для початківців. Khan Academy

пропонує широкий спектр навчальних матеріалів, включаючи лекції та вправи, з акцентом на доступність контенту.

Попри різноманіття існуючих ресурсів, дослідження показують, що не всі з них достатньою мірою забезпечують інтеграцію теорії з практикою, що є критично важливим для формування повноцінних навичок програмування. Часто користувачі зіштовхуються з фрагментарністю навчального матеріалу або недостатньою кількістю практичних завдань, що знижує ефективність засвоєння знань.

Крім того, більшість популярних платформ орієнтовані переважно на англomовну аудиторію, що може ускладнювати процес навчання для тих користувачів, які володіють англійською мовою на базовому рівні. Це створює додатковий бар'єр для залучення широкої аудиторії до процесу навчання.

Водночас слід зазначити позитивні тенденції у розвитку навчальних веб-систем, пов'язані зі зростанням популярності адаптивного навчання, інтерактивних симуляторів, гейміфікації навчального процесу та застосування елементів штучного інтелекту для персоналізації освітнього контенту. Такі підходи сприяють підвищенню мотивації користувачів і забезпечують індивідуалізований підхід до засвоєння матеріалу.

Таким чином, аналіз сучасного стану розвитку веб-систем для навчання основам веб-програмування дає змогу зробити висновок про наявність значного потенціалу для подальшого вдосконалення подібних рішень. Незважаючи на велику кількість існуючих ресурсів, існує потреба у створенні нових програмних продуктів, які поєднуюватимуть доступність, інтерактивність, достатній обсяг практичних завдань і враховуватимуть потреби користувачів різного рівня підготовки.

1.2. Порівняльний аналіз аналогів

На сучасному етапі розвитку інформаційних технологій значна кількість освітніх веб-ресурсів пропонує можливості для вивчення основ веб-програмування. Серед найбільш відомих та поширених платформ варто

виділити Codecademy, FreeCodeCamp та W3Schools. Кожна з них має свої особливості, переваги та недоліки, що зумовлює необхідність їхнього порівняльного аналізу для визначення доцільності використання в навчальному процесі.

Codecademy [3] є однією з провідних онлайн-платформ для вивчення програмування, яка пропонує структуровані курси з різних мов програмування, включаючи HTML, CSS та JavaScript. Основною особливістю цієї системи є інтерактивна форма навчання, коли користувачі безпосередньо в процесі вивчення матеріалу можуть писати код у вбудованому редакторі. Навчальні програми поділені на модулі, кожен з яких містить теоретичний матеріал і серію практичних завдань. Codecademy має частково безкоштовний доступ до курсів, проте повний функціонал (зокрема проекти, тести, практичні вправи підвищеної складності) відкривається лише після придбання платної підписки. Платформа також надає можливість слідкувати за навчальним прогресом, пропонує інтегровані завдання для самоконтролю та тестування, а також має активну спільноту користувачів, що дозволяє взаємодіяти з іншими студентами, ставити запитання та ділитися рішеннями. Codecademy продемонстрована на рисунку 1.1.

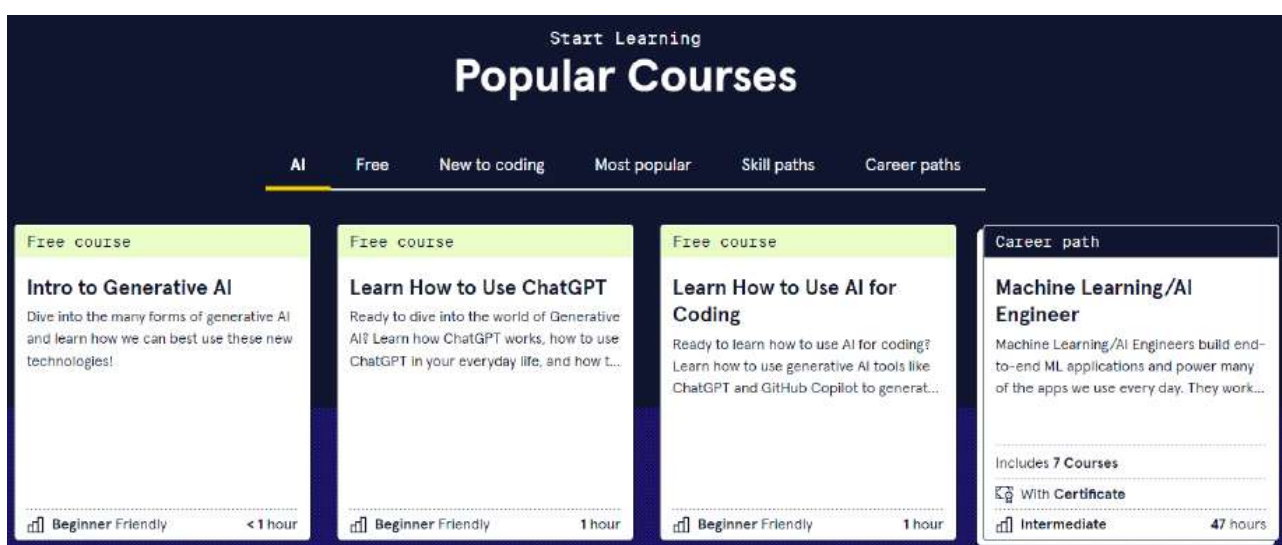


Рисунок 1.1 – Codecademy

FreeCodeCamp [5] позиціонує себе як повністю безкоштовний ресурс, орієнтований на самоосвіту. Платформа містить великий обсяг теоретичного матеріалу та практичних завдань, що охоплюють різні аспекти веб-розробки, починаючи з основ HTML і CSS і закінчуючи більш складними проєктами на JavaScript. Важливою перевагою FreeCodeCamp є наявність реальних практичних проєктів, які користувачі створюють у ході навчання. Питання для самоконтролю реалізовані в інтерактивній формі, що дає змогу перевірити розуміння матеріалу після кожного блоку. Крім того, платформа забезпечує фіксацію навчального прогресу, надаючи користувачам доступ до особистого кабінету зі статистикою успішності. Спільнота FreeCodeCamp є однією з найбільших серед освітніх ресурсів з програмування: існують форуми, групи у соціальних мережах, тематичні чати, що створює умови для взаємодопомоги та обміну досвідом між користувачами.

FreeCodeCamp показано на рисунку 1.2.

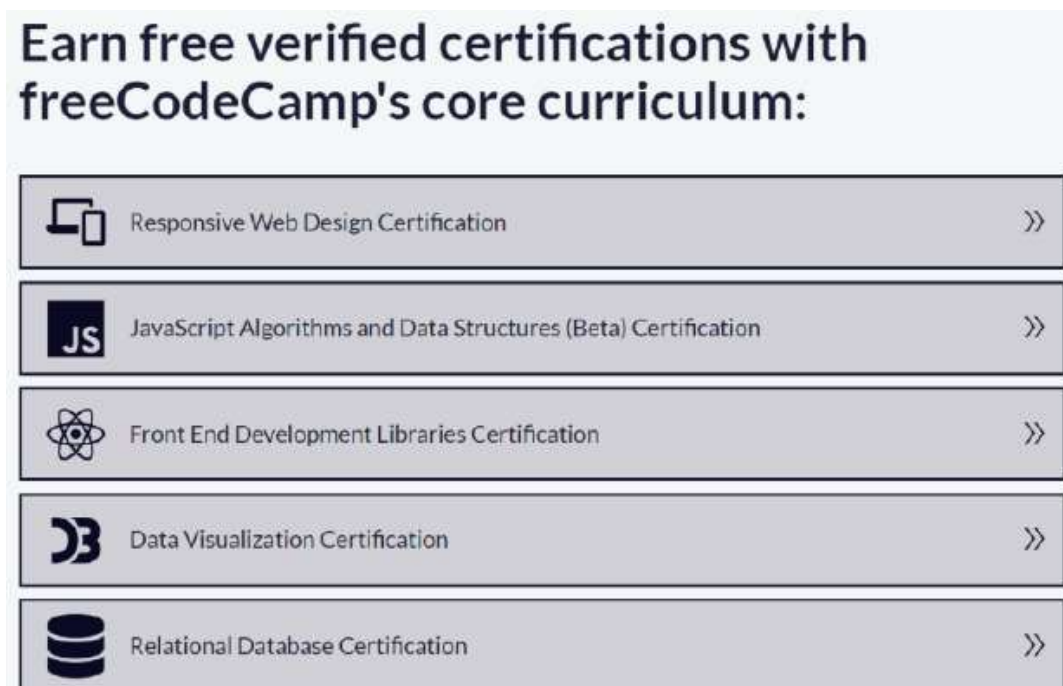


Рисунок 1.2 – FreeCodeCamp

W3Schools є одним із найстаріших і найвідоміших ресурсів для вивчення веб-технологій. Основна увага приділяється довідковій інформації, прикладам коду та коротким поясненням. Інтерактивна складова представлена можливістю

запускати приклади коду у спеціальному онлайн-редакторі (Try it Yourself), однак сама структура курсу менш інтерактивна, ніж на інших платформах. Доступ до основного контенту W3Schools є безкоштовним, проте для проходження сертифікаційних тестів необхідно оформити платну підписку. Щодо спільноти користувачів, W3Schools поступається конкурентам: хоча існують форуми та коментарі до матеріалів, рівень залученості користувачів є помітно нижчим. Практичні завдання представлені в обмеженій кількості, а питання для самоконтролю на платформі реалізовані переважно у вигляді тестів. Відслідковування прогресу частково доступне через особистий кабінет, але цей функціонал суттєво поступається можливостям Codecademy та FreeCodeCamp.

W3Schools показано на рисунку 1.3.

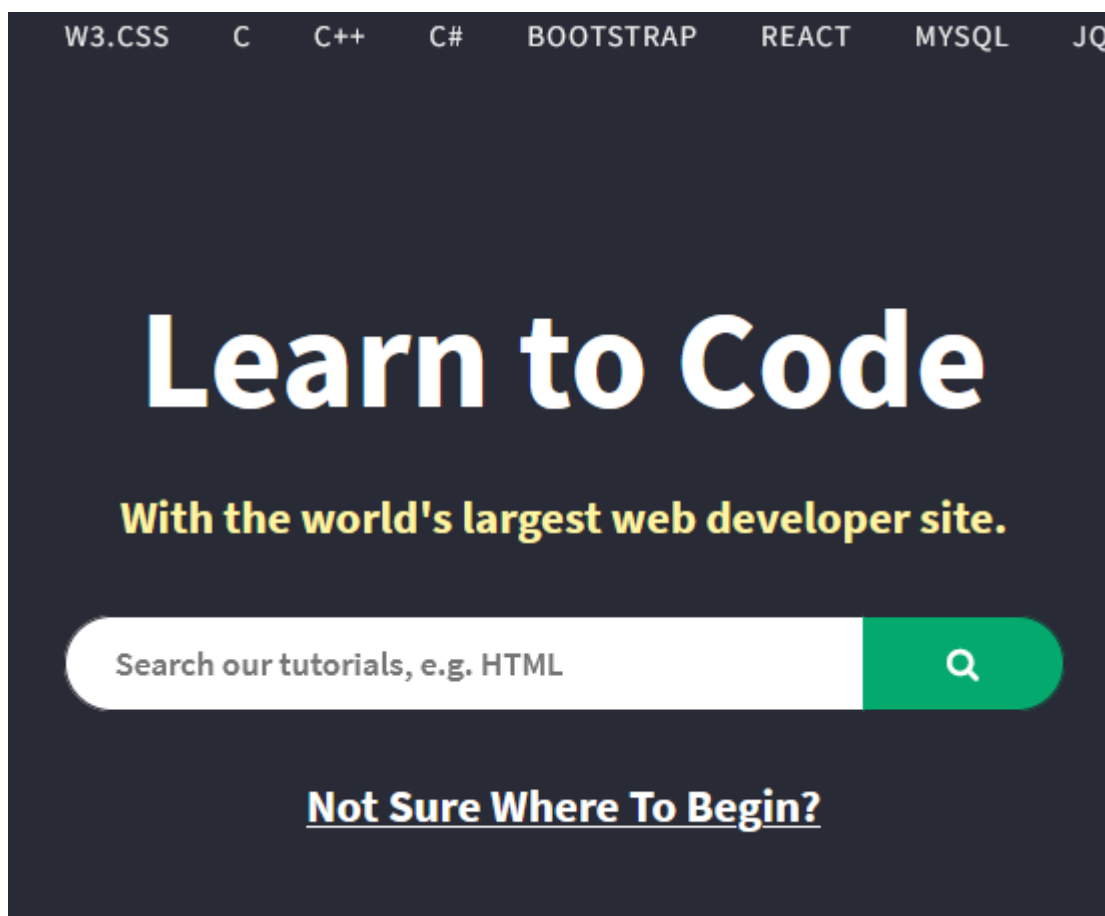


Рисунок 1.3 – W3Schools

Для наочності основні характеристики платформ наведено у таблиці (таблиця 1.1).

Таблиця 1.1 – Порівняння характеристик платформ

Характеристика	Codecademy	FreeCodeCamp	W3Schools
Безкоштовний доступ	Частково	Повністю	Частково
Спільнота користувачів	Активна	Дуже активна	Помірно активна
Практичні завдання	Є, інтерактивні	Є, реальні проекти	Обмежено
Питання для самоконтролю	Є	Є	Є (обмежено)
Відслідковування прогресу	Є	Є	Частково

Таким чином, проведений аналіз показує, що кожна з розглянутих платформ має як сильні, так і слабкі сторони. Codecademy вирізняється зручністю інтерфейсу, інтерактивністю і можливістю поступового контролю знань, однак має обмежений безкоштовний доступ. FreeCodeCamp є оптимальним рішенням з точки зору поєднання доступності, обсягу навчального матеріалу та залученості спільноти. W3Schools, у свою чергу, слугує корисним довідковим ресурсом, але менш ефективним як основна платформа для повноцінного навчання з нуля.

Зважаючи на проведене порівняння, можна зробити висновок про доцільність створення нових навчальних веб-додатків, які б поєднували інтерактивність, повноцінний безкоштовний доступ до матеріалів, наявність якісних практичних завдань та можливість ефективного відслідковування прогресу користувачів.

1.3. Постановка задачі

У результаті проведеного аналізу предметної області навчальних систем для вивчення основ веб-програмування, а також порівняльного дослідження існуючих аналогів і їх функціональних можливостей, було обґрунтовано доцільність розробки власної навчальної веб-системи. На основі вивчених методів розв'язання відповідних завдань визначено основні напрями

проектування, а також сформульовано перелік завдань, які необхідно реалізувати для досягнення поставленої мети.

Зокрема, передбачено проведення поглибленого аналізу наявних освітніх платформ для вивчення веб-програмування з метою виокремлення їх сильних і слабких сторін. На основі отриманих результатів планується розробити методичні засади, модель і алгоритмічне забезпечення функціонування навчальної системи з урахуванням потреб цільової аудиторії та актуальних тенденцій у галузі веб-розробки.

Функціональна архітектура розроблюваної системи передбачає створення базових компонентів веб-додатку, до яких належать механізми реєстрації та автентифікації користувачів, перегляд навчальних матеріалів, проходження тематичного тестування, інтерактивне середовище для написання і перевірки коду, а також модулі для адміністрування контенту – зокрема, додавання нових курсів і розробки їх змістового наповнення. Окремим етапом передбачається тестування системи на відповідність встановленим функціональним, технічним та ергономічним вимогам.

Структурно система включатиме кілька ключових модулів, інтегрованих у єдине інформаційне середовище. Зокрема, модуль управління даними відповідатиме за зберігання, обробку та організацію доступу до навчального контенту, тестів та результатів користувачів. Модуль графічного інтерфейсу забезпечуватиме зручний, інтуїтивно зрозумілий механізм взаємодії з системою для користувачів різних рівнів підготовки.

Відповідно до поставленої мети, проєктована веб-система повинна реалізувати такі функціональні можливості:

1. систему автентифікації та авторизації користувачів;
2. механізм адміністрування навчальних матеріалів і відеоконтенту;
3. інструменти для створення і розміщення питань самоконтролю;
4. можливість проходження тестів користувачами;
5. функціонал для відстеження динаміки навчального процесу та фіксації результатів тестування.

Така структура та функціональні характеристики навчальної веб-системи забезпечать ефективне опанування основ веб-програмування, сприятимуть систематизації знань та формуванню практичних навичок у користувачів.

РОЗДІЛ 2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ

2.1. Аналіз даних та розробка моделі системи

Дані та їх обробка становлять фундамент будь-якої сучасної програмної системи. У процесі функціонування інформаційних систем постійно генеруються значні обсяги інформації, що зумовлено як їх функціональним призначенням, так і потребами користувачів. Важливим джерелом даних є також інформація, що вводиться безпосередньо користувачами під час взаємодії з програмними системами задля реалізації своїх цілей.

Розроблювана навчальна система для вивчення основ веб-програмування не є винятком із цього правила. У процесі реєстрації користувачів система збирає та обробляє базовий набір персональних даних, необхідних для забезпечення функціонування сервісу. До таких даних належать ім'я користувача, адреса електронної пошти та пароль. Ім'я користувача використовується для ідентифікації особи в системі та формування відповідного запису у базі даних. Електронна адреса забезпечує можливість зв'язку з користувачем, зокрема для підтвердження реєстрації та надсилання службових повідомлень. Пароль слугує механізмом захисту облікового запису від несанкціонованого доступу.

З огляду на те, що процес навчання передбачає багатоетапне проходження навчального матеріалу, важливим є збереження проміжних результатів роботи користувача. Система фіксує інформацію про курси, які опановує користувач, а також його поточний прогрес, що дозволяє продовжувати навчання з моменту останнього входу. Усі ці дані зберігаються у відповідних структурах бази даних системи та прив'язуються до облікового запису користувача.

Після завершення вивчення курсу передбачається проходження тестування для перевірки рівня засвоєння матеріалу. Користувачеві пропонується опитувальник, відповіді на який автоматично зіставляються із заздалегідь

визначеними правильними варіантами. На основі цього порівняння система обчислює підсумковий результат тестування.

Окрім персональних даних користувачів, система також оперує даними про курси: їхній зміст, навчальні цілі, тривалість та структуровані навчальні матеріали, включаючи відеоконтент. Ці матеріали формують основу для організації навчального процесу й забезпечують якісне засвоєння теоретичного матеріалу.

Побудова моделі дозволяє формалізувати архітектуру системи, зробити її зрозумілою для всіх учасників процесу розробки, включно з розробниками, архітекторами та менеджерами.

Модель демонструє шляхи руху даних між окремими компонентами, зокрема Amazon Cognito, Active Directory, Amazon DocumentDB, Amazon API Gateway та іншими. Це сприяє глибшому розумінню логіки функціонування системи, її залежностей і потоків обробки інформації. Таким чином, модель виконує роль універсального інструменту комунікації та узгодження рішень у процесі проєктування.

Окрім цього, модель є основою для прийняття рішень щодо масштабування, модернізації чи заміни окремих компонентів у процесі подальшого розвитку системи. Вона надає цілісне уявлення про архітектуру, що є важливим для забезпечення стабільної, ефективної та безпечної роботи програмного продукту.

Структура моделі системи представлена на рисунку 2.1. До її основних компонентів належать:

1. Users – кінцеві користувачі системи.
2. Amazon Cognito – сервіс, що виконує функції автентифікації та авторизації користувачів.
3. Active Directory – система керування ідентифікацією користувачів та їх доступом.
4. Amazon DocumentDB – документно-орієнтована база даних, побудована на базі MongoDB.

5. Amazon API Gateway – шлюз для організації доступу до API сервісів системи.
6. AWS Lambda – серверлес-обчислення для виконання функціональних операцій.
7. Amazon CloudWatch – сервіс для моніторингу та адміністрування ресурсів AWS.
8. Amazon S3 – сховище об'єктів, яке використовується для збереження навчальних матеріалів, включно з мультимедійним контентом.

Завдяки інтеграції зазначених компонентів створюється функціонально завершена система, що відповідає сучасним вимогам до продуктивності, надійності та масштабованості веб-застосунків освітнього призначення.

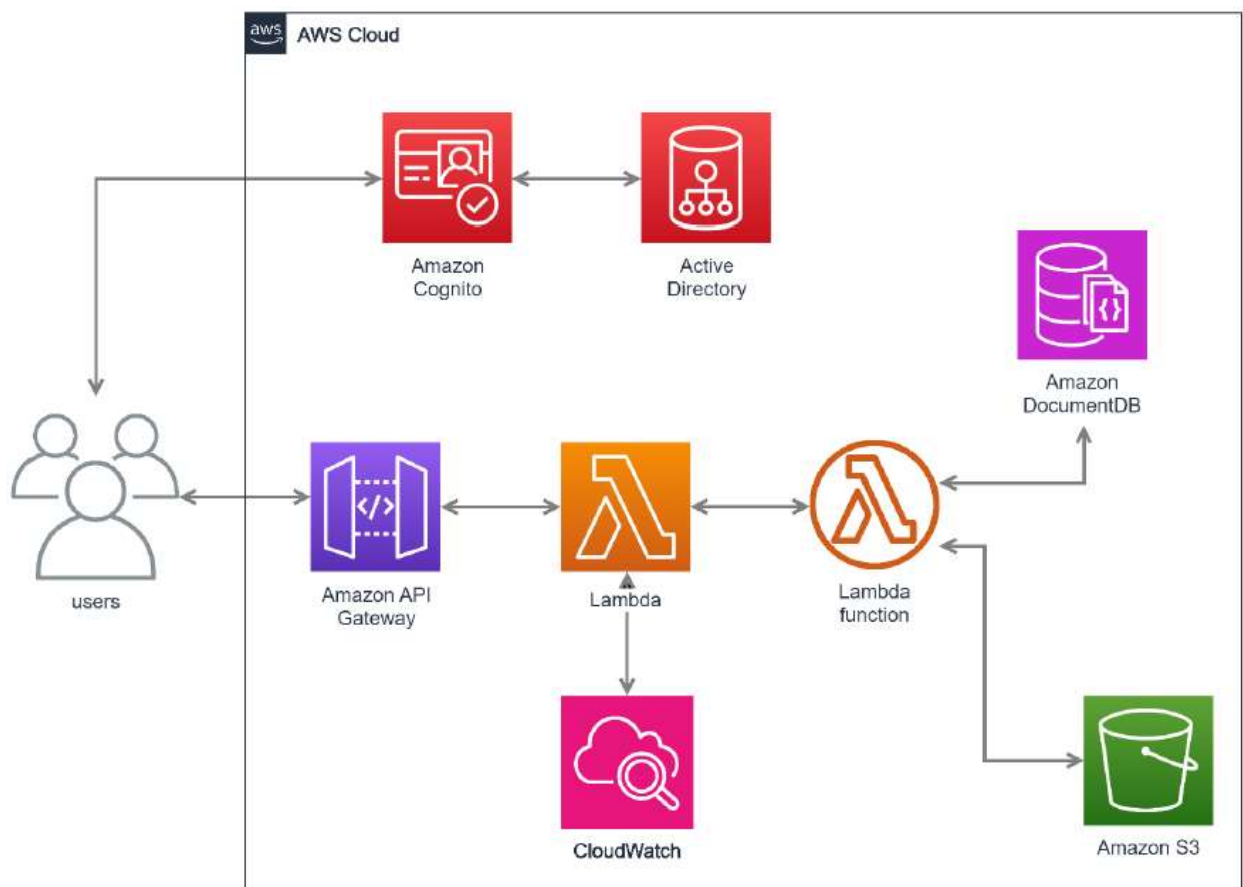


Рисунок 2.1 – Модель веб-застосунку

2.2. Розробка механізму керування навчальними курсами з урахуванням процедур автентифікації та авторизації користувачів

Додавання та редагування навчальних курсів адміністраторами є невід’ємною функціональною складовою веб-застосунку навчальної системи, орієнтованої на вивчення основ веб-програмування. Реалізація цієї функції забезпечує підтримку актуальності, повноти та якості навчального контенту, що надається користувачам. Водночас важливою складовою є механізм авторизації, що гарантує захист системи від несанкціонованого втручання у структуру навчальних матеріалів. Забезпечення належного рівня доступу передбачає розмежування прав користувачів, відповідно до призначених ролей у системі, що дозволяє уникнути несанкціонованого створення або редагування навчальних курсів. Авторизація здійснюється шляхом перевірки ролей користувачів, їх відповідності встановленим повноваженням, а також підтвердження приналежності курсів до конкретних авторизованих осіб.

Таким чином, система обмежує доступ до функцій створення та редагування курсів для звичайних користувачів, залишаючи ці можливості виключно за адміністраторами, викладачами, лекторами або авторами курсів. Такий підхід сприяє збереженню цілісності навчальних матеріалів і запобігає несанкціонованим змінам з боку осіб, які не є відповідальними за їх підготовку.

Загальна структура алгоритму реалізації процесів додавання та редагування курсів із використанням механізмів авторизації наведена на рисунку 2.2.

Методика реалізації процесів додавання та редагування курсів із забезпеченням авторизації включає декілька послідовних етапів:

На першому етапі здійснюється автентифікація користувача за допомогою методу *authenticate* класу *User*. Цей метод перевіряє облікові дані користувача (логін та пароль), які зберігаються у відповідній базі даних (*users_db*). У разі успішної перевірки користувач отримує доступ до функціональних можливостей системи.

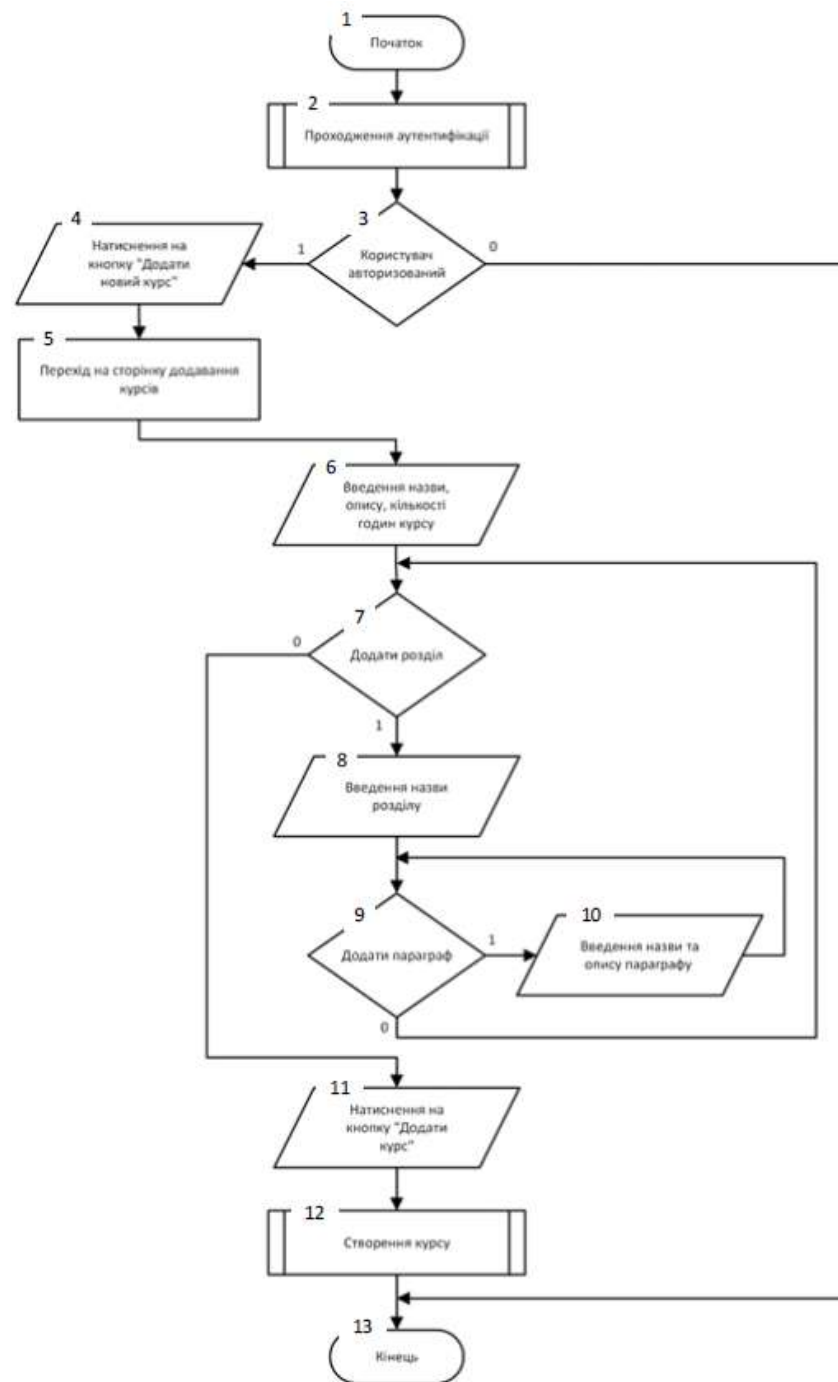


Рисунок 2.2 – Блок-схема алгоритму роботи методу додавання курсів

Далі автоматично виконується перевірка ролі користувача. Якщо користувач має роль *admin*, йому надаються права на створення нових курсів та внесення змін до вже існуючих.

Після цього користувач активує функцію створення нового курсу, що ініціює виклик методу *add_course* основної програми. Система переадресовує користувача на відповідний інтерфейс введення даних нового курсу.

На етапі введення інформації користувач заповнює необхідні поля: назву курсу, опис та тривалість у годинах. Ці дані тимчасово зберігаються у структурі об'єкта *course*.

Для створення змістовної структури навчального матеріалу додаються розділи, які формуються через виклик методу *add_section* класу *Course*. До кожного розділу додаються параграфи за допомогою методу *add_paragraph*, які містять назву та опис відповідного підрозділу.

Кожен параграф доповнюється тестовими завданнями, що створюються шляхом виклику методу *add_question*. Кожне тестове питання має кілька варіантів відповідей, серед яких визначається правильний, позначений спеціальним атрибутом.

Після завершення формування курсу здійснюється його збереження. Для цього користувач активує функцію *save_courses*, яка зберігає всі введені дані у форматі JSON у файлі *courses.json*. Збереження здійснюється через HTTP-запит типу POST до відповідного API-ендпоінту */courses*.

2.3. Розробка загальних алгоритмічних рішень для забезпечення функціонування застосунку

Загальний алгоритм функціонування програмного застосунку окреслює послідовність основних етапів та операцій, які здійснюються в процесі його роботи. Він визначає концептуальну структуру й логіку взаємодії користувача із системою, а також забезпечує цілісне уявлення про організацію процесів на високому рівні абстракції.

Алгоритм охоплює ключові сценарії взаємодії, зокрема процедури реєстрації та автентифікації користувачів, які виконуються залежно від наявності облікового запису. Подальший вибір користувача визначає перехід до відповідних функціональних модулів системи. Окремо передбачено обробку запитів, формування відповідей, здійснення взаємодії із системою керування базами даних та виконання серверних сценаріїв, що сукупно забезпечує

стабільність, ефективність і відповідність функціоналу потребам кінцевих користувачів.

Візуалізоване представлення загального алгоритму функціонування навчальної системи для вивчення основ веб-програмування наведено на рисунку 2.3.

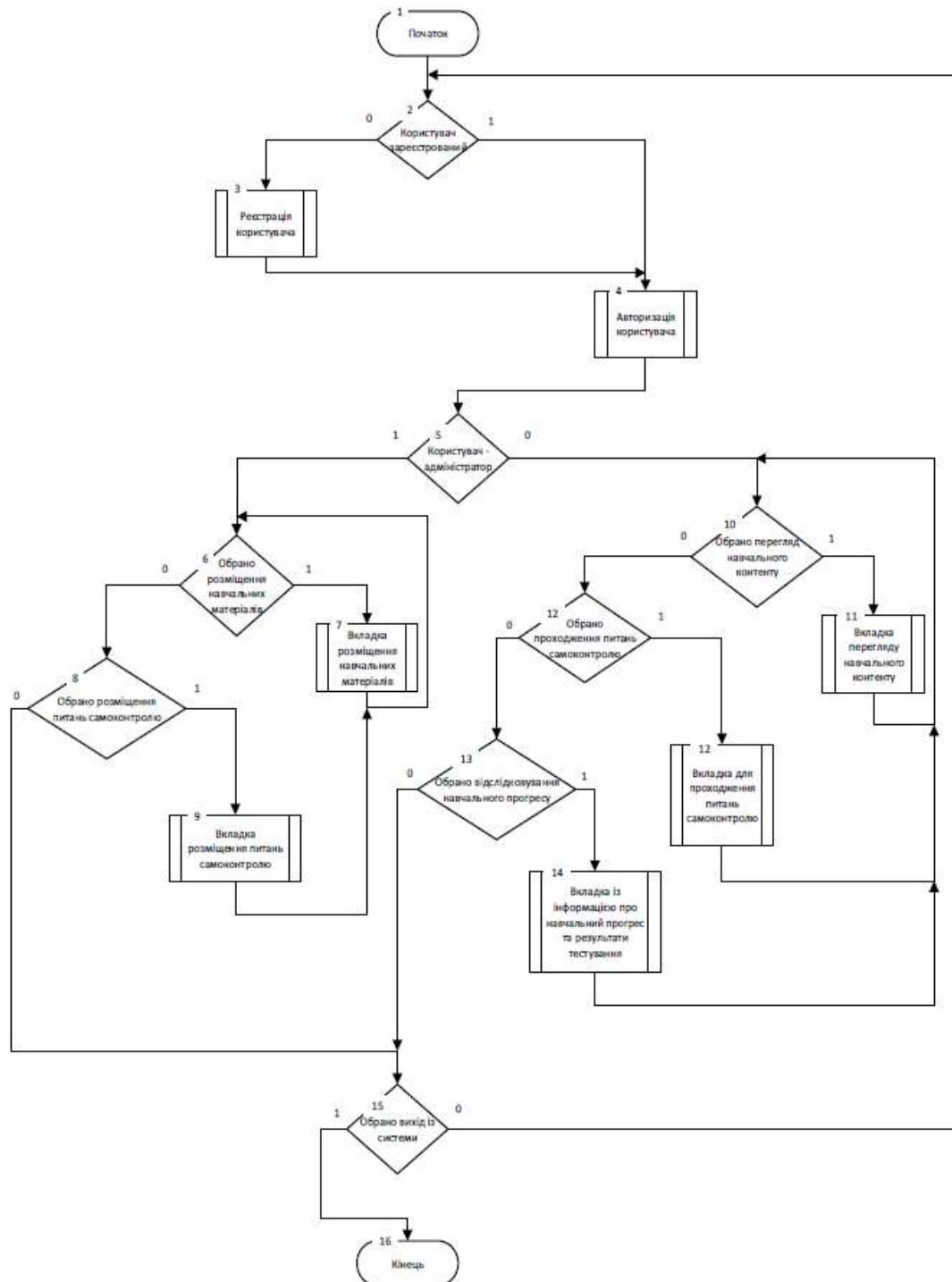


Рисунок 2.3 – Графічна модель алгоритмічної структури застосунку

Процес реєстрації є початковим етапом взаємодії користувача з системою, який передуює доступу до функціональних можливостей веб-застосунку. Для створення облікового запису користувачеві необхідно пройти процедуру реєстрації, що передбачає введення логіну та пароля.

Реєстрація здійснюється у кілька послідовних етапів. На першому етапі користувач заповнює форму реєстрації, зазначаючи персональні дані, серед яких ім'я, адресу електронної пошти, логін та пароль. Як правило, передбачено повторне введення пароля з метою перевірки коректності введених даних.

Наступним кроком є вибір унікального логіну, що надалі використовуватиметься для ідентифікації користувача у системі, а також формування пароля, який повинен відповідати встановленим вимогам інформаційної безпеки – зокрема, мати мінімальну довжину, містити цифри, великі літери та спеціальні символи. Схематичне відображення алгоритму реєстрації користувача представлено на рисунку 2.4.

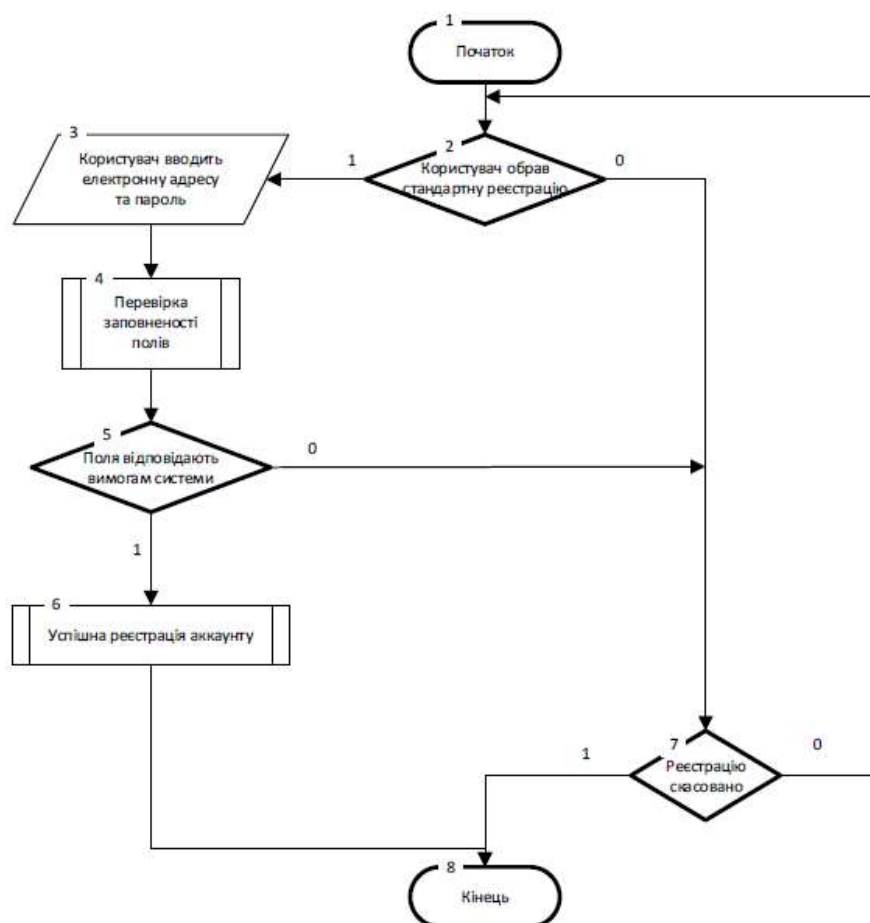


Рисунок 2.4 – Схематичне відображення алгоритму реєстрації користувача

Після завершення процедури реєстрації користувач переходить до етапу авторизації. Авторизація являє собою процес підтвердження автентичності, під час якого користувач вводить свої облікові дані – адресу електронної пошти та пароль – з метою отримання доступу до ресурсів системи. На цьому етапі система здійснює перевірку відповідності введених даних інформації, збереженій у базі даних, що гарантує захист від несанкціонованого доступу.

Схематичне зображення алгоритму процесу авторизації представлено на рисунку 2.5.

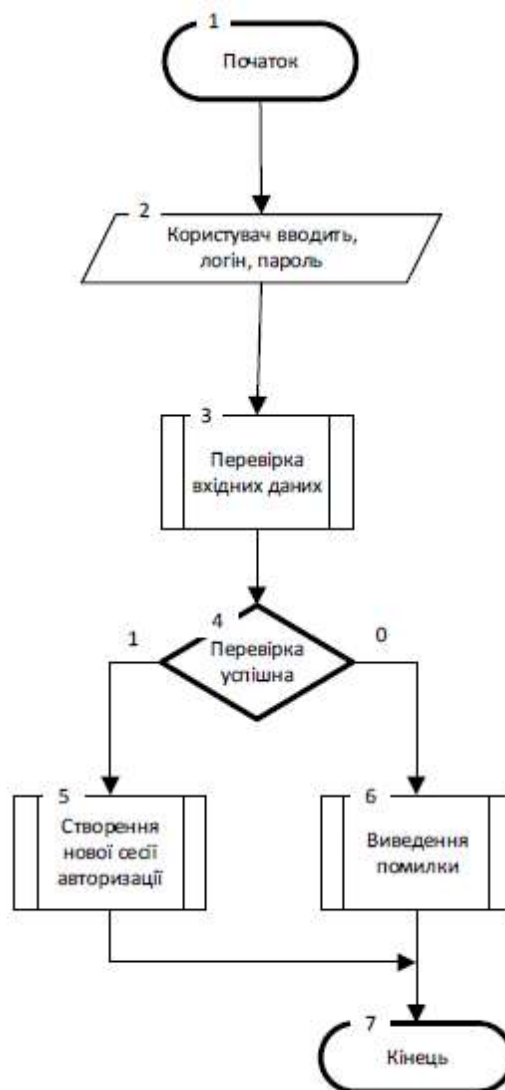


Рисунок 2.5 – Схематичне зображення алгоритму процесу авторизації

2.4. Аналіз та обґрунтування вибору мови програмування

Під час розробки програмних систем надзвичайно важливим етапом є вибір мови програмування, оскільки саме вона визначає архітектуру рішення, технологічний стек, особливості розробки, супроводу та масштабування продукту. Для створення веб-системи, призначеної для вивчення основ веб-програмування, доцільним є використання мови програмування, яка забезпечуватиме гнучкість у реалізації функціоналу, ефективну роботу з веб-технологіями та підтримку як клієнтської, так і серверної частини системи.

У ході аналізу різних мов програмування було розглянуто кілька варіантів: Python, PHP[11], Java, C# [12] та JavaScript. Кожна з цих мов має свої переваги, однак найбільш доцільним вибором у контексті розробки навчальної веб-системи виявився саме JavaScript. Основним аргументом такого вибору стала його універсальність: JavaScript є єдиною мовою, яка може однаково ефективно застосовуватись як для фронтенду, так і для бекенду.

JavaScript [10] – це високорівнева, інтерпретована мова програмування, яка з моменту свого створення стала основою для побудови інтерактивних веб-сторінок. Вона підтримується всіма сучасними веб-браузерами, що робить її незамінною для клієнтської частини веб-додатків. Крім того, завдяки появі середовища виконання Node.js JavaScript [13] успішно застосовується і на стороні сервера. Це забезпечує можливість реалізувати повноцінний повностековий підхід до розробки, коли весь програмний код написаний однією мовою, що істотно спрощує процес інтеграції, підтримки та розвитку системи.

На клієнтському рівні JavaScript використовується для:

- реалізації інтерактивних елементів веб-сторінки;
- динамічного оновлення вмісту без перезавантаження сторінки;
- обробки подій (наприклад, натискання кнопок, введення тексту, вибору елементів);
- взаємодії з сервером через API (зокрема, за допомогою технологій Fetch API або бібліотеки Axios);

- реалізації сучасних інтерфейсів за допомогою фреймворків React, Vue.js або Angular.

На серверному рівні за допомогою Node.js JavaScript забезпечує[14]:

- створення REST API для обробки запитів від клієнта;
- роботу з базами даних (наприклад, MongoDB через бібліотеку Mongoose);
- реалізацію механізмів аутентифікації та авторизації користувачів;
- обробку бізнес-логіки застосунку;
- забезпечення масштабованості завдяки асинхронній моделі обробки запитів.

Серед інших переваг JavaScript варто зазначити:

- Популярність: за даними різних рейтингів (зокрема Stack Overflow Developer Survey), JavaScript багато років поспіль займає перші місця серед мов програмування за популярністю.
- Велика спільнота: активна спільнота розробників забезпечує доступ до великої кількості навчальних ресурсів, бібліотек і готових рішень, що значно полегшує розробку.
- Масштабованість: JavaScript легко інтегрується з хмарними рішеннями (наприклад, Amazon Web Services, Google Cloud Platform, Microsoft Azure), що дозволяє масштабувати систему в майбутньому.
- Гнучкість: існує великий вибір інструментів і бібліотек для розробки як backend, так і frontend частин системи.
- Простота інтеграції: використання єдиної мови програмування для клієнтської і серверної частини мінімізує складність взаємодії між різними компонентами застосунку.

Зважаючи на те, що навчальна система передбачає активну взаємодію користувачів через веб-інтерфейс, динамічну генерацію контенту, підтримку автентифікації, авторизації, можливість виконання інтерактивних завдань, тестування знань тощо, застосування JavaScript є повністю виправданим. Це дозволяє досягти високого рівня інтерактивності, оптимізувати обробку даних у реальному часі та забезпечити ефективне управління навчальним процесом.

Отже, вибір JavaScript як основної мови програмування для реалізації навчальної системи для вивчення основ веб-програмування є логічно обґрунтованим з позицій як технічної доцільності, так і стратегічного бачення розвитку проекту. Це рішення сприяє ефективній реалізації поставлених завдань, підтримує сучасні підходи до розробки та забезпечує перспективи подальшого вдосконалення і масштабування системи.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ

3.1. Реалізація модуля автентифікації та реєстрації користувачів

Модуль автентифікації, реалізований у розробленій веб-системі для вивчення основ веб-програмування, спроектовано з урахуванням забезпечення користувачам інтуїтивно зрозумілого та ефективного механізму реєстрації й входу до системи. Після запуску веб-застосунку користувачеві відображається головна сторінка, на якій передбачено можливість вибору між створенням нового облікового запису та авторизацією за допомогою наявних реєстраційних даних (див. рисунок 3.1).

The image displays two side-by-side web forms for user authentication and registration. The left form, titled 'Реєстрація' (Registration), contains three input fields labeled 'Введіть Ім'я' (Enter Name), 'Email', and 'Пароль' (Password). Below these fields is a link that reads 'У Вас вже є акаунт? Увійти' (Do you already have an account? Log in) and a blue button labeled 'Створити акаунт' (Create account). The right form, titled 'Вхід' (Login), contains two input fields labeled 'Email' and 'Пароль' (Password), followed by a blue button labeled 'Увійти' (Log in).

Рисунок 3.1 – Зображення структури інтерфейсу сторінок автентифікації та реєстрації користувачів

Модуль автентифікації забезпечує процес ідентифікації користувачів у системі шляхом введення облікових даних. Основними його елементами є форма авторизації, яка передбачає введення електронної пошти та пароля. Після надсилання даних формується запит до серверної частини застосунку, де виконується їх перевірка. У разі успішної верифікації користувач отримує доступ до персоналізованого середовища системи та відповідного функціоналу.

Форма авторизації реалізована з використанням засобів фреймворку Vue.js [15]. Для організації двостороннього зв'язку між елементами інтерфейсу та внутрішніми даними компонента застосовується директива 'v-model'. Це забезпечує синхронізацію введених користувачем даних із відповідними

змінними у логіці компонента. Після підтвердження введених даних активується метод 'axios.post' [16], який здійснює передачу інформації на сервер у форматі JSON-об'єкта [17]. У тілі запиту передаються значення електронної пошти та пароля, відповідно до полів форми.

На серверній стороні виконується процес валідації введених даних. У разі відповідності наданої інформації записам у базі даних MongoDB [18], сервер формує авторизаційний токен і повертає його клієнту. Цей токен використовується для організації сеансу користувача та надання доступу до захищених частин веб-застосунку.

У випадку успішної автентифікації (HTTP-статус 200) реалізується автоматичне перенаправлення користувача до головного інтерфейсу системи за допомогою методу 'this.\$router.push('/')'. При отриманні негативної відповіді (наприклад, статус 401 [19], що вказує на помилку авторизації) користувачеві виводиться відповідне повідомлення з поясненням причини відмови у доступі (див. рисунок 3.2).

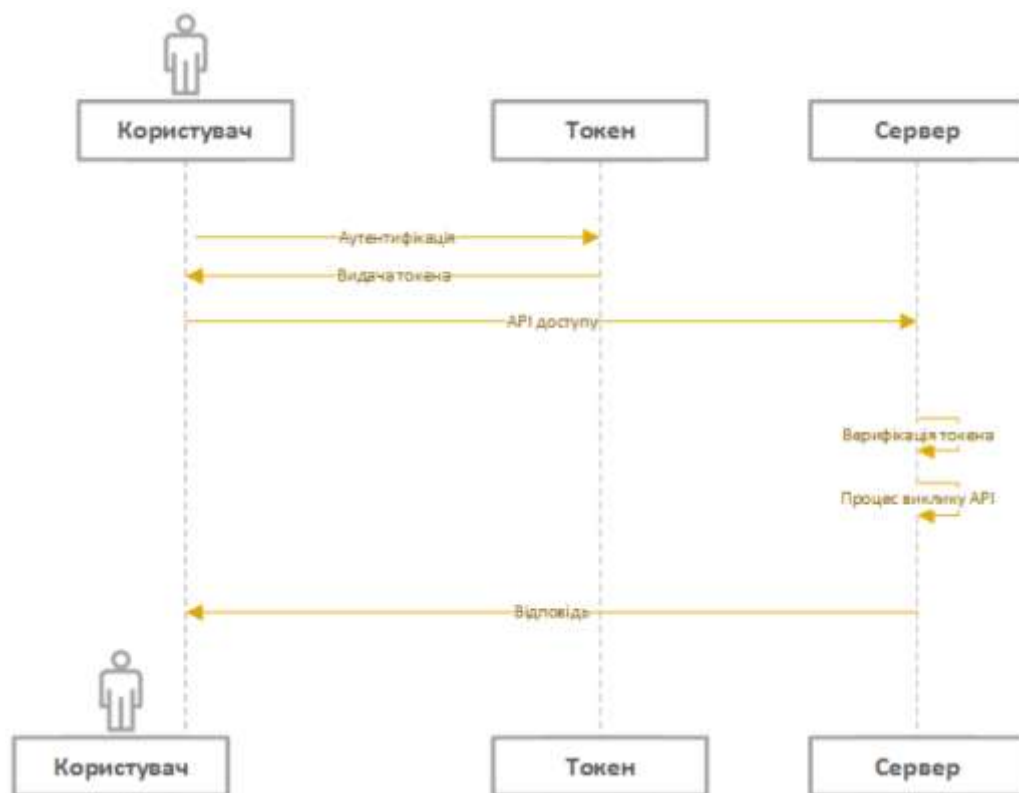


Рисунок 3.2 – Діаграма послідовності процесу автентифікації користувача

Процес реєстрації нових користувачів у системі передбачає створення персонального облікового запису для подальшої автентифікації та використання функціональних можливостей веб-застосунку. Для цього користувачеві необхідно заповнити відповідну форму, зазначивши персональні дані, зокрема ім'я, адресу електронної пошти та пароль. Ця інформація надалі зберігається на сервері й використовується для ідентифікації користувача під час авторизації.

Форма для введення реєстраційних даних реалізована із застосуванням фреймворку Vue.js, де механізм двостороннього зв'язування 'v-model' забезпечує синхронізацію стану даних між елементами інтерфейсу та змінними компонента. Завдяки цьому введені значення автоматично відображаються у відповідних змінних програми.

Після підтвердження введених даних користувачем відправка запиту здійснюється через метод 'axios.post', який формує HTTP-запит до сервера у форматі JSON. Цей запит містить основні параметри: ім'я користувача, адресу електронної пошти та пароль.

На сервері отримані дані проходять процедуру валідації. У разі успішної перевірки здійснюється створення нового запису у відповідній колекції бази даних. За результатами обробки сервер формує відповідь, яка містить інформацію про успішне завершення операції.

Після отримання позитивної відповіді застосунок автоматично здійснює перенаправлення користувача на сторінку автентифікації ('this.\$router.push('/login)'), де користувач може увійти до системи, використовуючи щойно створені облікові дані. У разі виникнення помилок (наприклад, спроба реєстрації з уже зареєстрованою електронною адресою) користувач отримує відповідне повідомлення з поясненням причини відмови в реєстрації.

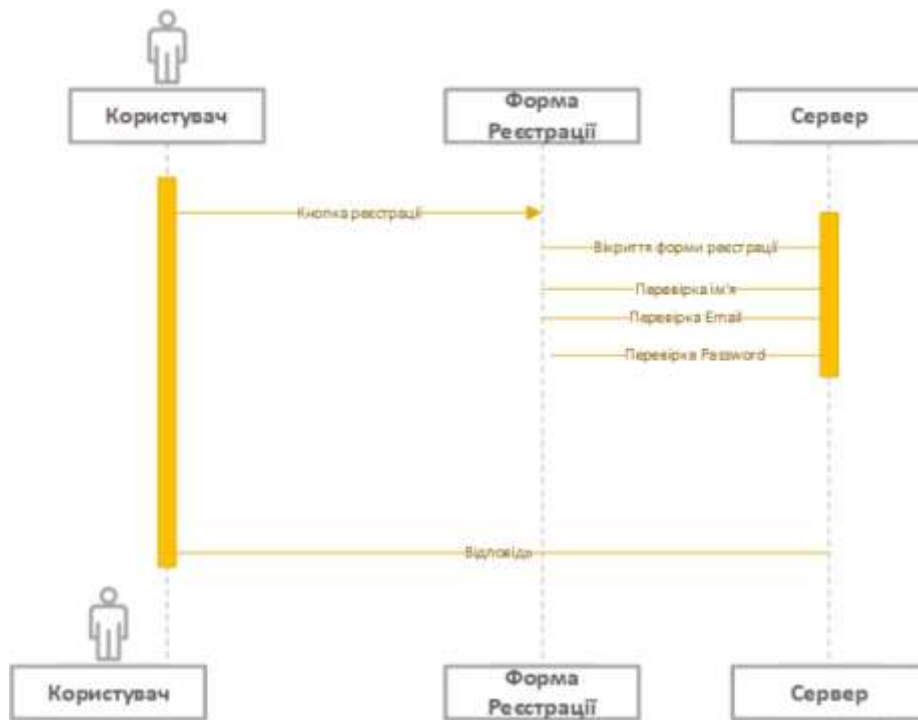


Рисунок 3.3 – Послідовність процесу реєстрації користувача

Для забезпечення взаємодії користувачів із базою даних системи було спроектовано відповідну структуру об'єктів, сформовану відповідно до їхніх ролей у системі. Відповідна схема цієї структури представлена на рисунку 3.4.

```

"users": [
  {
    "id": "dicom5_",
    "name": "SD",
    "email": "merh@gmail.com",
    "password": "123qwerty",
    "roles": [
      "ADMIN",
      "USER"
    ]
  },

```

Рисунок 3.4 –Клас для забезпечення взаємодії користувачів

Характеристика об'єкта:

- id – унікальний ідентифікатор запису, що забезпечує однозначну ідентифікацію кожного користувача в системі;
- name – особисте ім'я користувача, необхідне для персоналізації взаємодії з системою;

- email – електронна адреса, що використовується для реєстрації, автентифікації та зворотного зв'язку;
- password – захищений елемент, який гарантує безпечний доступ до персонального облікового запису;
- roles – набір ролей, які визначають рівень доступу користувача до функціональних можливостей системи.

Наведена структура забезпечує ефективне керування обліковими записами, дозволяючи користувачам проходити автентифікацію, а також отримувати доступ до відповідних навчальних матеріалів відповідно до їхніх повноважень.

3.2. Реалізація модуля для формування навчальних курсів

Однією з ключових функціональних складових навчальних веб-систем є забезпечення можливості розміщення навчальних матеріалів. Реалізацію цієї функції забезпечує модуль створення курсів.

Користувач, наділений відповідними правами доступу, має можливість створювати нові курси, а також редагувати ті, які належать йому як автору. Для ініціації процесу створення курсу необхідно здійснити авторизований вхід до системи, після чого стає доступною функціональна кнопка «Додати новий курс», призначена для зареєстрованих користувачів із необхідними правами (див. рисунок 3.5).

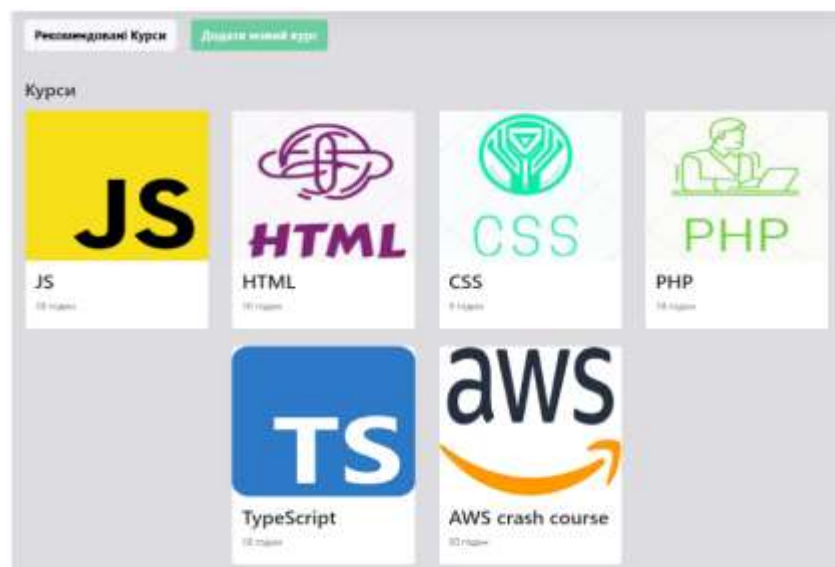


Рисунок 3.5 – Процес створення навчального курсу

Після активації відповідної кнопки користувач перенаправляється на сторінку з інтерактивною формою, яка призначена для введення необхідної інформації про курс. На цій сторінці користувач має можливість додати опис курсу, завантажити зображення, сформувати розділи та наповнити їх відповідними параграфами (див. рисунок 3.6).

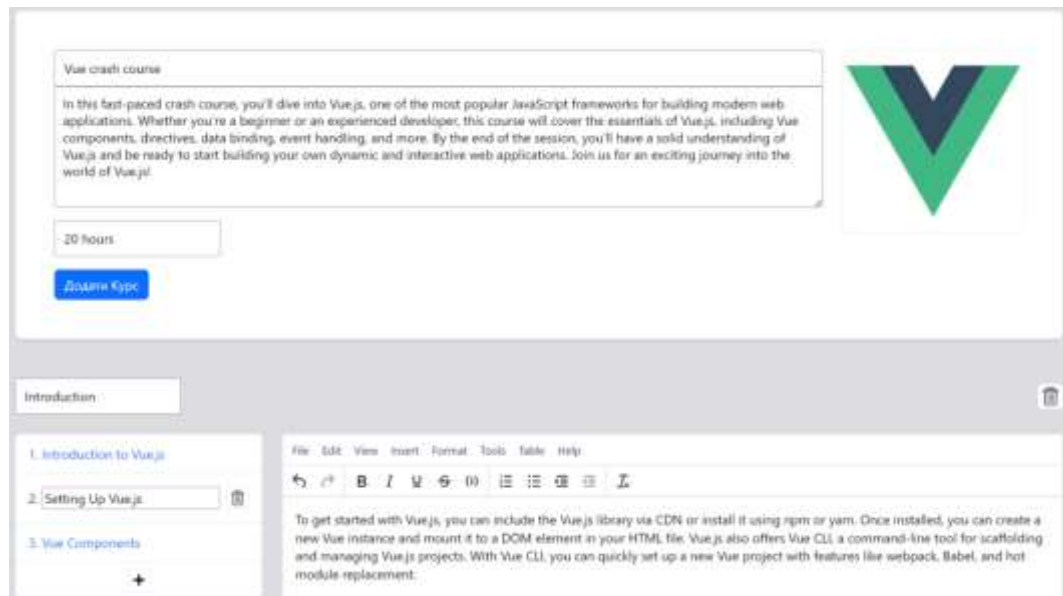


Рисунок 3.6 – Формування навчального курсу

Далі користувач може зберегти створений курс шляхом натискання кнопки «Додати курс». У результаті виконується HTTP POST-запит [20] до сервера, який передає дані курсу у форматі JSON для збереження в базі даних (див. рисунок 3.7).

```
* POST http://localhost:3000/courses
* Network
* Request Headers
  Content-Type: "application/json"
  User-Agent: "PostmanRuntime/7.37.3"
  Accept: "*/*"
  Postman-Token: "cee73883-2b99-4661-9e37-1917b66e288e"
  Host: "localhost:3000"
  Accept-Encoding: "gzip, deflate, br"
  Connection: "keep-alive"
  Content-Length: "2378"
* Request Body
  {
    "name": "Vue crash course",
    "description": "In this fast-paced crash course, you'll dive into Vue.js, one of the most popular JavaScript frameworks for building modern web applications. Whether you're a beginner or an experienced developer, this course will cover the essentials of Vue.js, including Vue components, directives, data binding, event handling, and more. By the end of the session, you'll have a solid understanding of Vue.js and be ready to start building your own dynamic and interactive web applications. Join us for an exciting journey into the world of Vue.js!",
    "duration": "20 hours",
    "courseImageUrl": "/img/Vue.png",
    "chapters": [
      {
        "title": "Introduction",
        "paragraphs": [
          {
            "title": "Introduction to Vue.js",
            "text": "Vue.js is a progressive JavaScript framework for building user interfaces. It is designed from the ground up to be incrementally adoptable, meaning you can start using it for small parts of your application and gradually scale"
          }
        ]
      }
    ]
  }
```

Рисунок 3.7 – Виконання запиту додавання курсу

У результаті виконання запиту формується відповідний запис у базі даних у форматі JSON (див. рисунок 3.8).



Рисунок 3.8 – Структура збереженого курсу в базі даних

На рисунку 3.9 представлено збережений курсу на головній сторінці веб-додатку.

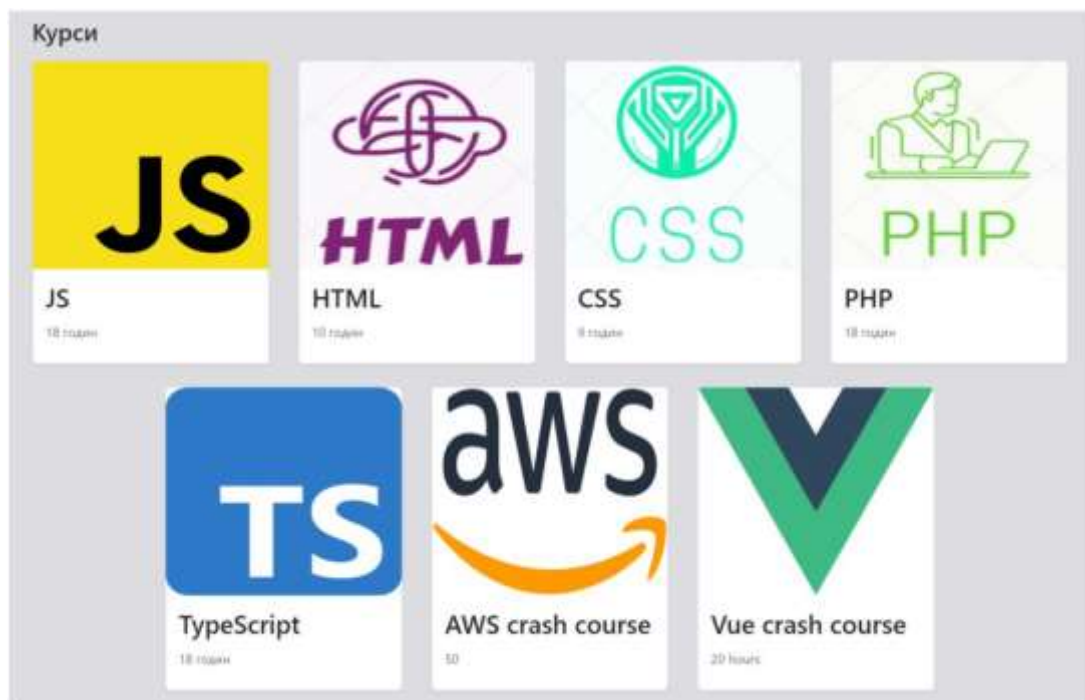


Рисунок 3.9 – Демонстрація збереженого курсу на головній сторінці

3.3. Реалізація модуля для проходження тестування

Модуль проходження тестування є одним із основних елементів освітніх вебплатформ, онлайн-курсів та систем оцінювання, оскільки забезпечує можливість перевірки знань і навичок користувачів у певній предметній області. Основне призначення цього модуля полягає в організації процесу тестування, що включає отримання запитань, введення відповідей та оцінювання результатів.

Функціональна структура модуля реалізована у вигляді компонента `TestForm.vue`, який відповідає за формування інтерфейсу тестування та відображення всіх запитань. Кожне окреме запитання оформлено як компонент `Question.vue`, що дозволяє забезпечити модульність і гнучкість інтерфейсу. Після завершення тестування результати користувача відображаються за допомогою компонента `Result.vue`, який відповідає за візуалізацію підсумкової інформації.

Для забезпечення роботи модуля реалізовано низку методів:

- `fetchQuestions()` – викликається для отримання переліку запитань з сервера. Для цього використовується HTTP-запит методом GET, здійснений за допомогою бібліотеки `Axios`.
- `submitAnswers()` – використовується для передачі відповідей користувача на сервер. Здійснюється HTTP-запит методом POST, у якому дані передаються у форматі JSON.
- `calculateScore()` – відповідає за обчислення підсумкового результату тестування на основі кількості правильних відповідей. Реалізація алгоритму залежить від специфіки системи оцінювання.

Процес взаємодії з сервером реалізовано через `Axios`, який надсилає HTTP-запити до відповідних API endpoint для отримання запитань та надсилання відповідей.

Типовий сценарій проходження тестування складається з таких етапів:

1. Ініціація тестування шляхом виклику методу `fetchQuestions()`, що забезпечує завантаження переліку запитань з серверної частини застосунку.
2. Введення відповідей користувачем у відповідних полях компонентів `Question.vue`.
3. Надсилення відповідей на сервер за допомогою методу `submitAnswers()`.
4. Обробка відповідей на сервері, обчислення підсумкових результатів і передача цих результатів клієнту.
5. Відображення результатів у компоненті `Result.vue`, що забезпечує завершення процесу тестування.

Таким чином, модуль тестування забезпечує повний цикл взаємодії користувача із системою – від отримання завдань до отримання результатів, реалізуючи інтуїтивно зрозумілий та ефективний механізм перевірки знань.

```
// src/components/TestForm.vue
<template>
  <div>
    <h2 class="text-xl font-bold mb-4">Пройходження тестування</h2>

    <div v-if="loading" class="text-gray-500">Завантаження запитань...</div>

    <div v-else>
      <Question
        v-for="(question, index) in questions"
        :key="question.id"
        :question="question"
        v-model="answers[question.id]"
      />

      <button @click="submitAnswers" class="mt-4 px-4 py-2 bg-blue-500 text-white rounded">Завершити тест</button>

      <Result v-if="result !== null" :score="result" />
    </div>
  </div>
</template>

<script>
import axios from 'axios'
import Question from './Question.vue'
import Result from './Result.vue'

export default {
  components: { Question, Result },
  data() {
    return {
      questions: [],
      answers: {},
      result: null,
      loading: false,
    }
  }
}
```

```

    }
  },
  methods: {
    async fetchQuestions() {
      this.loading = true
      try {
        const response = await axios.get('/api/questions')
        this.questions = response.data
      } catch (error) {
        console.error('Помилка завантаження запитань:', error)
      } finally {
        this.loading = false
      }
    },
    async submitAnswers() {
      try {
        const response = await axios.post('/api/submit', this.answers)
        this.result = response.data.score
      } catch (error) {
        console.error('Помилка надсилання відповідей:', error)
      }
    },
  },
  mounted() {
    this.fetchQuestions()
  },
}
</script>

<style scoped>
button:hover {
  background-color: #2563eb;
}
</style>

```

Цей модуль формує гнучке та інтерактивне середовище для проходження тестування, що дає змогу користувачам оперативно відстежувати власний прогрес і отримувати зворотний зв'язок щодо рівня засвоєння матеріалу або сформованих навичок.

3.4. Тестування додатку для опанування основ веб-програмування

Тестування є ключовим етапом розробки веб-застосунків, що забезпечує відповідність системи визначеним вимогам і коректність її функціонування. Процес тестування охоплює декілька послідовних етапів, зокрема планування, підготовку, виконання тестів, а також аналіз результатів. На етапі планування визначаються цілі та обсяг тестування, формуються тестові сценарії та обираються відповідні інструменти. Підготовка включає налаштування середовища й створення тестових даних. Виконання тестування охоплює

функціональні, юзабіліті, продуктивні та суміснісні перевірки. Завершальним етапом є аналіз результатів, складання звітності та формування рекомендацій щодо вдосконалення системи. У процесі можуть застосовуватися як ручні, так і автоматизовані методи тестування. На рисунку 3.11 представлено процес перевірки головної сторінки системи, де демонструється відображення рекомендованих курсів відповідно до уподобань користувача.

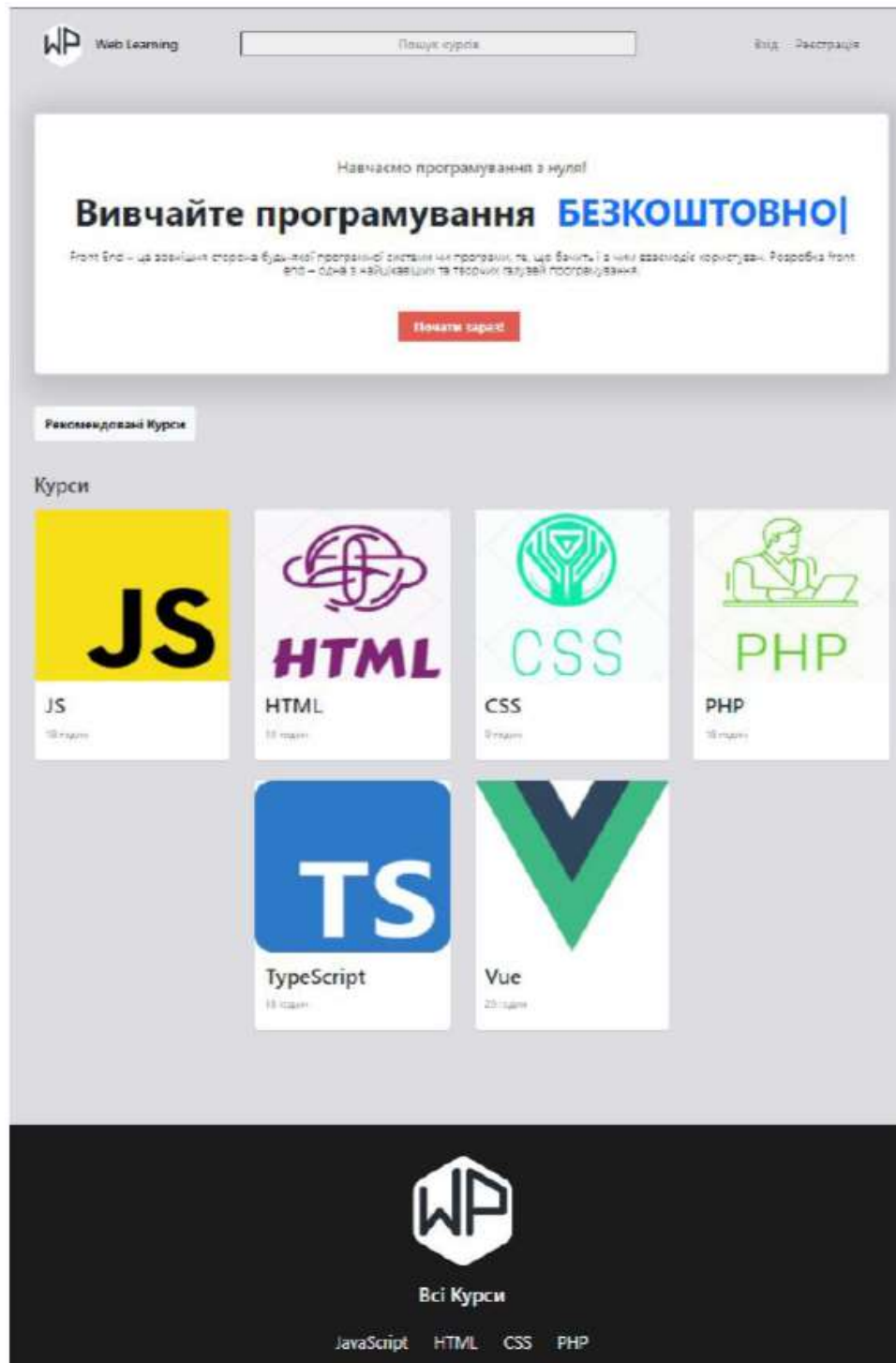


Рисунок 3.11 – Тестування головної сторінки додатку

На рисунку 3.12 представлено процес тестування функціональності пошуку. У наведеному прикладі виконано пошуковий запит за ключовими словами «Веб програмування». У відповідь система формує та відображає список курсів, що відповідають тематиці створення користувацьких інтерфейсів.

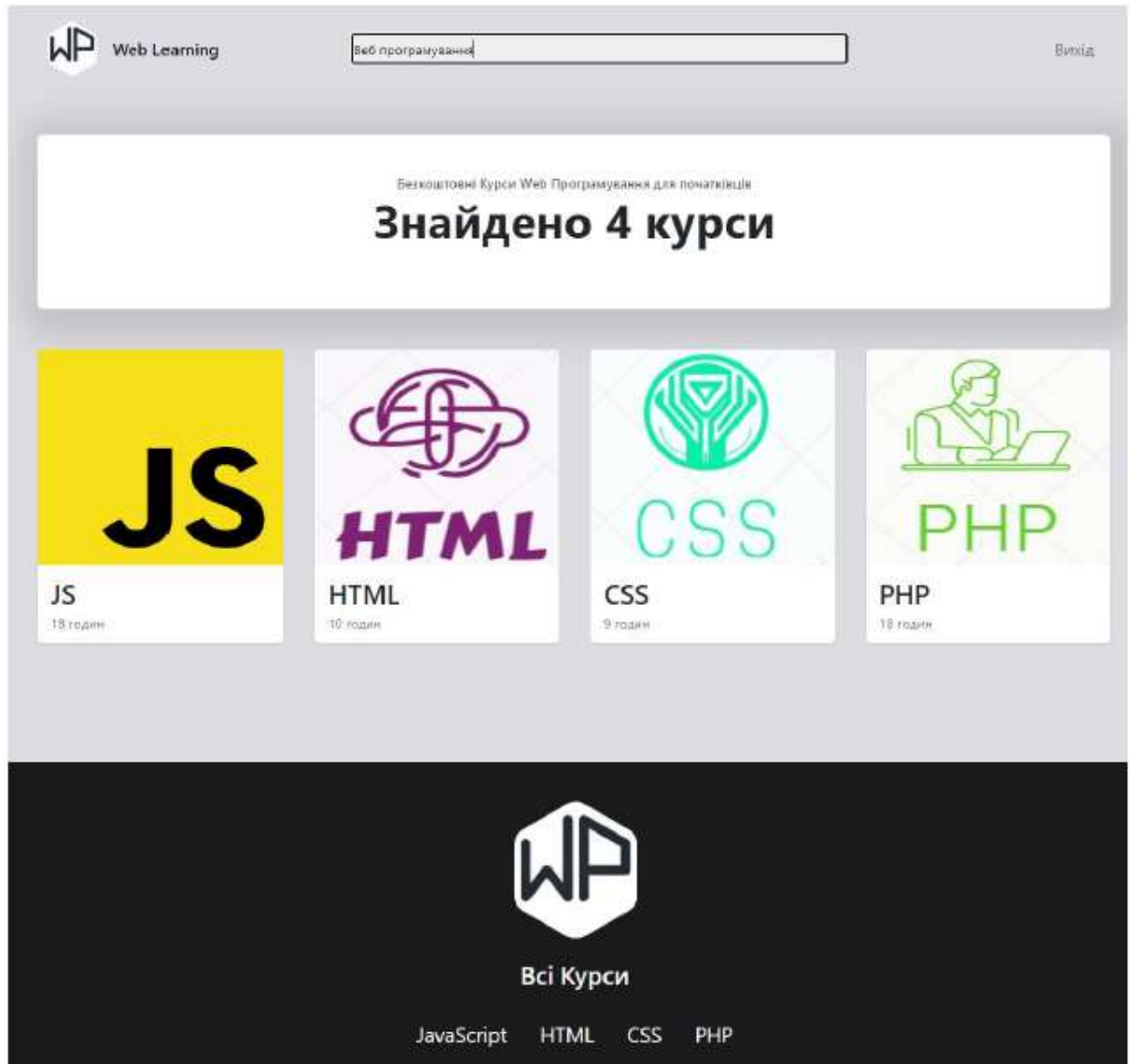


Рисунок 3.12 – Тестування функціоналу пошуку навчальних курсів

Для прикладу було обрано один із доступних курсів зі списку. Після натискання на відповідний елемент інтерфейсу здійснюється перехід на сторінку перегляду курсу, де відображається його назва, розгорнутий опис та елемент керування для початку навчання. Результати тестування відображення інформації про курс наведено на рисунку 3.13.

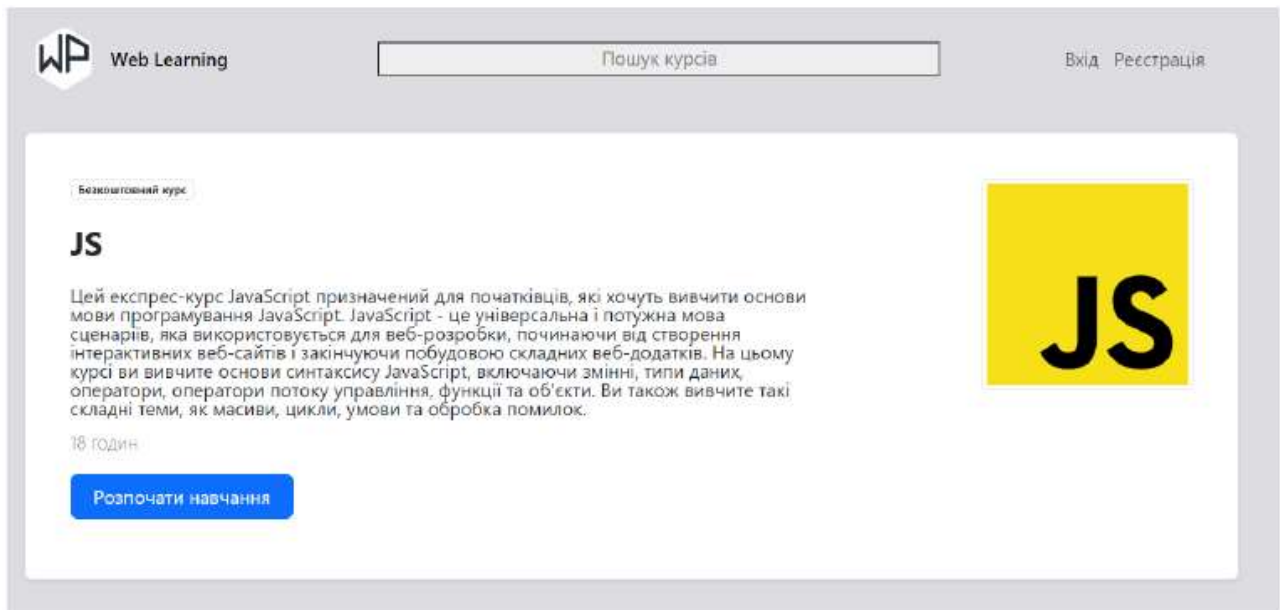


Рисунок 3.13 – Відображення детальної інформації про вибраний навчальний курс

Після проходження процедури авторизації користувачу відкривається повний доступ до навчальних матеріалів обраного курсу. Система забезпечує можливість навігації між темами та перегляду їхнього змісту у зручному форматі. Результати тестування функціоналу перегляду структури курсу представлено на рисунку 3.14.



Рисунок 3.14 – Розділ перегляду обраного курсу

Після завершення вивчення курсу користувачу надається можливість перевірити набуті знання та навички. Для цього передбачено окремий інтерфейс, що містить дві частини: у лівій області розміщується тест із варіантами відповідей, а також практичне завдання, яке передбачає написання програмного коду для розв’язання поставленої задачі. Приклад вигляду вікна тестування представлено на рисунку 3.15.

Перевірка засвоєння матеріалу за темою HTML

Тестування

Виберіть один із варіантів відповіді:

1) Що таке семантичний HTML і чому він важливий?

- ☐ Використання HTML тегів для стилізації сторінки
- ☐ Використання HTML коментарів для пояснення коду
- ☒ Використання нестандартних тегів для підвищення SEO

2) Що таке атрибут data-* в HTML?

- ☐ Використовується для зберігання користувацьких даних на HTML елементах
- ☐ Використовується для визначення типу даних введення в форму
- ☒ Використовується для встановлення тексту інструментів

3) За допомогою якого тега можна створити посилання у документі?

- ☐ <p>
- ☐ <link>
- ☒ <a>

Практичне Завдання

Завдання

Розробіть веб-сторінку для невеликого блогу, використовуючи сучасні стандарти HTML5. Веб-сторінка повинна включати наступні елементи: семантичну структуру (використовуйте семантичні теги), розмістіть елементи на сторінці у логічній послідовності; навігаційне меню у верхній частині сторінки з посиланнями на різні розділи блогу; одну статтю блогу з заголовком, текстом, зображенням та підзаголовками; поле введення для пошуку по сайту; форму з полями введення для коментарів, що включає перевірку введення email; вбудоване відео; нумерований список корисних ресурсів; таблицю з розкладом публікацій; атрибути "data-*" для зберігання додаткової інформації на HTML елементах; адаптивний дизайн для різних пристроїв за допомогою медіа-запитів у CSS; простий SVG елемент, наприклад, логотип блогу.

1 |

[Перевірити](#)



Всі Курси

JavaScript HTML CSS PHP

Рисунок 4.5 – Перевірка засвоєння матеріалу

Інтерфейс створення курсу для адміністратора призначений для додавання нових навчальних курсів до системи. Адміністратору надається доступ до форми з полями для введення назви курсу, його короткого опису, тривалості навчання та завантаження зображення (логотипу) курсу.

Крім того, передбачена можливість формування структури курсу шляхом створення окремих розділів і додавання до них параграфів, що забезпечує логічний поділ навчального матеріалу.

Зовнішній вигляд інтерфейсу додавання курсу наведено на рисунку 3.16.

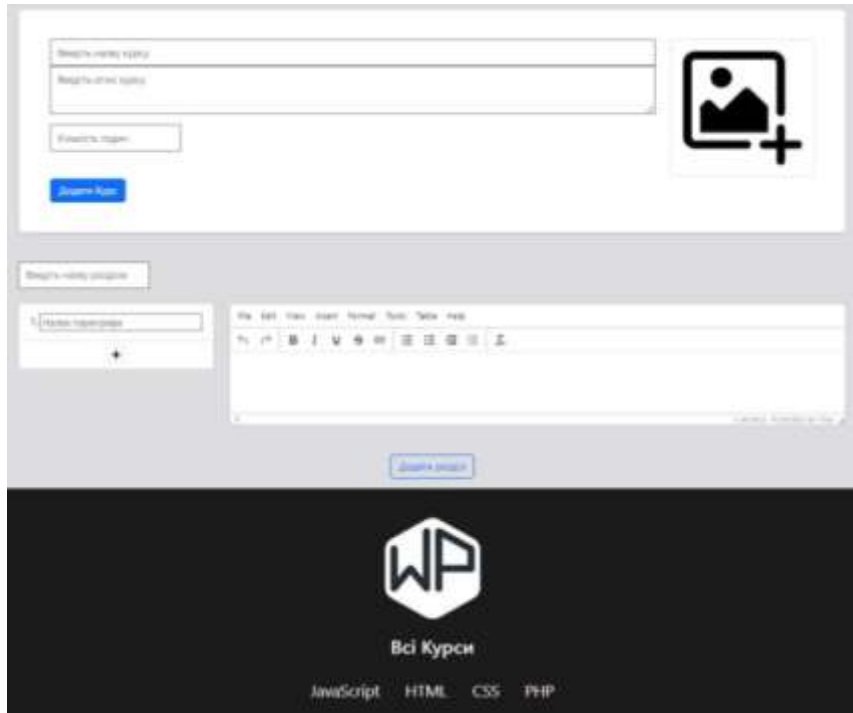


Рисунок 3.16 – Вигляд інтерфейсу додавання курсу

Інтерфейс редагування курсу за структурою аналогічний до форми створення курсу, однак відрізняється тим, що в поля форми автоматично підставляються раніше збережені дані про вибраний курс. Адміністратор має можливість змінювати наявну інформацію відповідно до потреб. Вигляд інтерфейсу редагування курсу представлено на рисунку 3.17.

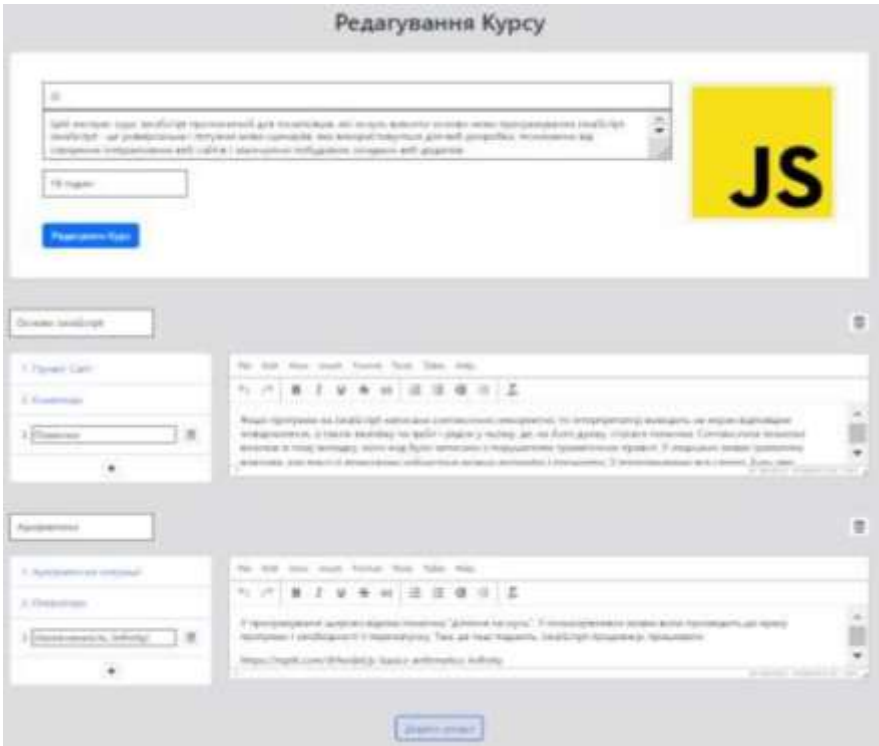


Рисунок 3.17 – Вигляд інтерфейсу редагування

Інформаційне вікно користувача забезпечує доступ до персональних даних, збережених у системі, відомостей про призначені ролі та переліку курсів, у яких бере участь користувач. Крім того, відображається прогрес засвоєння навчальних матеріалів. Приклад інтерфейсу інформаційного вікна користувача показано на рисунку 3.18.

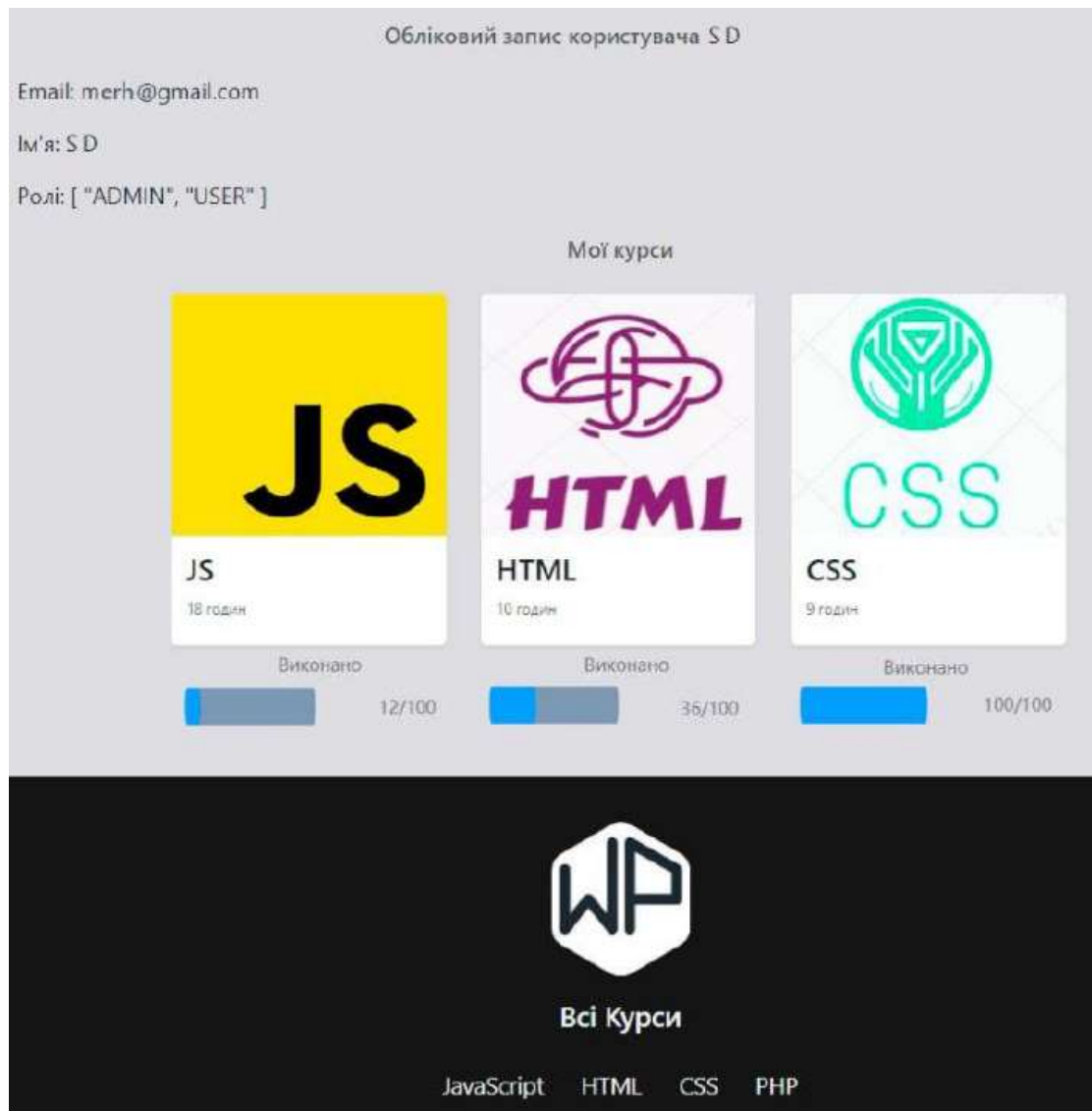


Рисунок 3.18 - Відомості про користувача

Отже, було здійснено тестування веб-додатку для вивчення основ веб-програмування, у межах якого продемонстровано основні функціональні можливості. Результати тестування підтвердили, що розроблена система успішно реалізує поставлені на початковому етапі проектування цілі та завдання.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було реалізовано веб-орієнтований додаток, призначений для підтримки процесу вивчення основ веб-програмування. В результаті проведеного дослідження було сформовано цілісний підхід до створення навчального середовища, що забезпечує користувачам доступ до навчальних матеріалів, можливість проходження тестування та отримання результатів, які відображають рівень опанування матеріалу.

Під час реалізації проєкту було детально проаналізовано предметну область, що дозволило визначити основні вимоги до функціональності системи та обрати найбільш доцільні інструменти для її створення. Застосування сучасних веб-технологій, таких як Vue.js для клієнтської частини та Node.js у поєднанні з базою даних MongoDB для серверної частини, забезпечило необхідний рівень продуктивності, масштабованості та зручності у використанні. Особливу увагу приділено реалізації інтерактивного інтерфейсу, що сприяє ефективній взаємодії користувача із системою.

Структура програмного забезпечення побудована таким чином, щоб охоплювати всі ключові етапи навчального процесу – від реєстрації користувача, доступу до навчальних курсів до можливості складання тестів і перегляду результатів. Реалізована система дозволяє авторизованим користувачам створювати власні навчальні курси, доповнювати їх структурованими матеріалами, а також організовувати перевірку знань за допомогою тестових завдань. Це дає змогу адаптувати процес навчання під індивідуальні потреби кожного користувача, що підвищує ефективність засвоєння матеріалу.

У результаті проведеного тестування було підтверджено відповідність реалізованої системи заявленим функціональним вимогам. Система продемонструвала стабільну роботу при виконанні всіх основних сценаріїв використання, коректне функціонування механізмів збереження та обробки даних, а також належний рівень інтерактивності користувацького інтерфейсу.

Окремо варто зазначити ефективність реалізованого модуля тестування, який забезпечує не лише перевірку знань, але й надає користувачеві зворотний зв'язок, що є важливим елементом у процесі навчання.

Таким чином, розроблена веб-система відповідає поставленим завданням та демонструє практичну доцільність для застосування у навчальних процесах. Перспективи подальшого розвитку проєкту передбачають удосконалення функціональних можливостей системи, розширення інструментів взаємодії з користувачем, інтеграцію адаптивних технологій навчання, а також підвищення рівня персоналізації освітнього процесу відповідно до індивідуальних потреб кожного користувача.

СПИСОК ЛІТЕРАТУРИ

1. Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.
2. Crockford, D. (2008). JavaScript: The Good Parts. O'Reilly Media.
3. Resig, J., & Bibeault, B. (2013). Secrets of the JavaScript Ninja (2nd ed.). Manning Publications.
4. Freeman, E., & Robson, E. (2020). Head First JavaScript Programming. O'Reilly Media.
5. Zakas, N. C. (2016). Understanding ECMAScript 6. No Starch Press.
6. Zakas, N. C. (2021). Professional JavaScript for Web Developers (4th ed.). Wrox.
7. McFarland, D. S. (2015). JavaScript & jQuery: The Missing Manual (3rd ed.). O'Reilly Media.
8. Duckett, J. (2014). JavaScript and jQuery: Interactive Front-End Web Development. Wiley.
9. Tuck, J. (2020). Modern JavaScript for Beginners. Independently published.
10. Svensson, A. (2021). JavaScript from Beginner to Professional. Amazon KDP.
11. Valenzuela, M. (2020). ES6 and Beyond: Understanding Modern JavaScript. Leanpub.
12. Rauschmayer, D. (2019). Speaking JavaScript: An In-Depth Guide for Programmers. O'Reilly Media.
13. Rauschmayer, D. (2021). Exploring ES6: Upgrade to the Next Version of JavaScript. Leanpub.
14. Myers, G. (2019). JavaScript: Functional Programming for JavaScript Developers. Packt Publishing.
15. Karpov, V. (2021). Mastering Async/Await in JavaScript. The Code Barbarian.
16. JetBrains. (2024). WebStorm Documentation. Retrieved from <https://www.jetbrains.com/webstorm/documentation/>
17. Moore, C. (2020). WebStorm Essentials. Packt Publishing.
18. JetBrains. (2023). Getting Started with WebStorm. JetBrains Blog.

- 19.DiMarzio, J. (2022). Using JetBrains WebStorm for Full-Stack Development. *Journal of Web Tools*, 8(2), 43–59.
- 20.Smith, R. (2021). Boosting JavaScript Productivity with WebStorm. *DevPro Magazine*, 12(4), 15–23.
- 21.Zhang, Y. (2022). Advanced Debugging Techniques in WebStorm. *CodeCraft Journal*, 17(1), 50–64.
- 22.Johnson, P. (2023). Mastering WebStorm Shortcuts and Workflows. *JetBrains Insider*.
- 23.Sbarski, P., & Kroonenburg, S. (2020). *Serverless Applications with AWS*. Manning Publications.
- 24.Holcombe, J. (2022). *AWS Certified Developer Guide*. Sybex.
- 25.Wilkins, B. (2021). *Getting Started with AWS Cognito*. Apress.
- 26.Cope, A. (2022). *User Authentication with Amazon Cognito*. Packt Publishing.
- 27.Kavis, M. J. (2014). *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. Wiley.
- 28.Herrera, A. (2023). Securing Web Applications with AWS IAM and Cognito. *CloudSec Journal*, 14(3), 22–39.
- 29.Date, C. J. (2021). *An Introduction to Database Systems (8th ed.)*. Addison-Wesley.
- 30.Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems (7th ed.)*. Pearson.
- 31.Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts (7th ed.)*. McGraw-Hill.
- 32.Ullman, J. D., & Widom, J. (2008). *A First Course in Database Systems (3rd ed.)*. Pearson.
- 33.Coronel, C., & Morris, S. (2022). *Database Systems: Design, Implementation, & Management (14th ed.)*. Cengage.
- 34.Han, J., Pei, J., & Kamber, M. (2021). *Data Mining: Concepts and Techniques (4th ed.)*. Morgan Kaufmann.

- 35.Linstedt, D., & Olschimke, M. (2015). Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann.
- 36.Malik, S. (2023). AWS for Web Developers: S3, CloudFront, Route53. Developer Academy Series.
- 37.Deshmukh, A. (2022). Securing Data in S3 Buckets: Best Practices. Cloud Security Review, 18(1), 61–73.
- 38.Smith, J. (2021). Data Storage Patterns on AWS: S3, RDS, and Beyond. AWS DevOps Magazine, 9(3), 24–38.

ДОДАТКИ