Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ТЕСТУВАННЯ /ОПИТУВАННЯ В НАВЧАЛЬНОМУ ПРОЦЕСІ

INFORMATION SYSTEM FOR TESTING /INWESTIGATION IN THE EDUCATIONAL PROCESS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Шворак Артем Миколайович

(підпис)

Керівник:
к.т.н., доцент
Бортник Катерина Яківна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» червня 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: «*Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н. Чернящук
«_____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Швораку Артему Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інформаційна система для тестування /опитування в навчальному процесі.*

Керівник роботи: *Бортник Катерина Яківна, к.т.н., доцент*

затверджені наказом закладу вищої освіти від «28» грудня 2022 р. №982/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2023 р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Вступ
Огляд предметної області
Проектування веб-ресурсу для підприємства
Програмна реалізація та тестування роботи веб-ресурсу
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Використані технології: архітектура системи, структура серверної та клієнтської частин програмного комплексу, схеми об'єктів бази даних, блок-схема роботи програми,

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд предметної області	*Бортник К.Я.*		
Проектування інформаційної системи	*Бортник К.Я.*		
Програмна реалізація та тестування роботи веб-ресурсу	*Бортник К.Я.*		
Висновки	*Бортник К.Я.*		

7. Дата видачі завдання 01.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Обґрунтування теми*	12.11.2022	Виконано
2.	*Дослідження літератури*	20.11.2022	Виконано
3.	*Аналіз предметної області*	14.01.2023	Виконано
4.	*Обгрунтування технологій та засобів розробки*	02.02.2023	Виконано
5.	*Проектування веб-ресурсу*	25.02.2023	Виконано
6.	*Тестування роботи веб-ресурсу*	24.03.2023	Виконано
7.	*Оформлення матеріалів роботи*	15.05.2023	Виконано
8.	*Нормоконтроль*	25.05.2023	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	01.06.2023	Виконано
10.	*Представлення кваліфікаційної роботи бакалавра до захисту*	07.06.2023	Виконано

Здобувач вищої освіти _____ (Шворак А. М..)
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (Бортник К. Я.)
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Шворак А. М. Інформаційна система для тестування /опитування в навчальному процесі. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2023.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі здійснено огляд предметної області, розглядаються платформи і сервіси для перевірки знань.

Другий розділ присвячений розробці архітектури системи, проєктуванню структури серверної і клієнтської частин програмного комплексу.

У третьому розділі виконано перевірку роботи веб ресурсу, шляхом тестування.

Об'єкт дослідження – існуючі сучасні програмні рішення для організації та проведення тестів у навчальному процесі.

Предмет дослідження – веб ресурс Web Mentor для організації та проведення тестів у навчальному процесі, побудований на основі класичної трирівневої архітектури, яка включає клієнтську частину, серверну частину та базу даних.

Метою роботи є розроблення веб орієнтованого ресурсу для організації та проведення тестування й опитувань у навчальному процесі.

Ключові слова: інформаційна система, веб ресурс, інтерфейс програми, тестування.

ABSTRACT

Shvorak A. Information system for testing / survey in the educational process. Manuscript.

Qualification work of the bachelor of the OP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2023.

Qualification work consists of an introduction, three sections, conclusions, a list of sources used.

The first section provides an overview of the subject area, platforms and services for testing knowledge are considered.

The second section is devoted to the development of the system architecture, the design of the structure of the server and client parts of the software complex.

The third section checks the operation of the web resource by testing.

The object of the study is existing modern software solutions for organizing and conducting tests in the educational process.

The subject of the study is the Web Mentor web resource for organizing and conducting tests in the educational process, built on the basis of a classic three-tier architecture, which includes a client part, a server part and a database.

The aim of the work is to develop a web-based resource for organizing and conducting testing and surveys in the educational process.

Keywords: information system, web resource, program interface, testing.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій веб ресурси стали невід'ємною складовою освітнього процесу. Вони не лише забезпечують швидкий і зручний доступ до навчальних матеріалів, а й суттєво розширюють можливості контролю та перевірки рівня знань здобувачів освіти. Застосування веб технологій у навчанні сприяє підвищенню якості засвоєння матеріалу та оптимізації організації освітньої діяльності.

Метою роботи є розроблення веб орієнтованого ресурсу для організації та проведення тестування й опитувань у навчальному процесі. У процесі виконання роботи використано сучасні веб технології, зокрема мови програмування Java Script та PHP, веб сервер Apache, систему керування базами даних MySQL, а також технологію асинхронного обміну даними AJAX.

Об'єкт дослідження – існуючі сучасні програмні рішення для організації та проведення тестів у навчальному процесі.

Предмет дослідження – веб ресурс Web Mentor для організації та проведення тестів у навчальному процесі, побудований на основі класичної трирівневої архітектури, яка включає клієнтську частину, серверну частину та базу даних.

Завдання, які необхідно виконати:

- здійснити огляд існуючих сучасних програмних рішень для організації та проведення тестів у навчальному процесі;
- розроблення архітектури системи для організації та проведення тестування й опитувань у навчальному процесі;
- проєктування алгоритму роботи програми;
- тестування веб ресурсу.

Вхідними даними системи є інформація про тести та опитування, що зберігається у базі даних, зокрема структура тестів, перелік запитань, варіанти відповідей і налаштування проходження. Вихідними даними мають бути вебсторінки у форматі HTML, на яких відображається інформація про доступні

тести або опитування та результати їх виконання. Результатом функціонування системи є збережені та впорядковані результати пройдених тестів і опитувань.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Мета та підхід до порівняльного аналізу

Аналіз сучасних систем перевірки знань є важливим і необхідним етапом у процесі проєктування власного веб ресурсу, призначеного для організації тестування та опитувань у навчальному процесі. В умовах активної цифровізації освіти заклади все частіше використовують різноманітні програмні рішення для контролю рівня знань здобувачів освіти. Такі системи відрізняються між собою за функціональними можливостями, складністю впровадження та супроводу, зручністю користування, а також орієнтацією на конкретні категорії користувачів – викладачів, студентів або адміністраторів.

У сучасному освітньому середовищі представлено значну кількість платформ і сервісів для перевірки знань, які можуть використовуватися як у межах традиційного очного навчання, так і в умовах дистанційної або змішаної форми освіти. Вибір конкретного програмного рішення значною мірою впливає на ефективність навчального процесу, об’єктивність оцінювання та зручність організації контролю знань.

Метою порівняльного аналізу є визначення сильних і слабких сторін найбільш поширених на ринку систем перевірки знань, а також узагальнення їх функціональних характеристик з метою формування вимог до перспективного веб ресурсу. Проведення такого аналізу дає змогу оцінити доцільність використання наявних рішень, виявити їх обмеження та визначити напрями вдосконалення систем тестування.

У рамках дослідження особлива увага приділяється таким аспектам, як рівень автоматизації процесів оцінювання, можливість підтримки дистанційного навчання, гнучкість налаштувань тестових завдань, аналітичні інструменти для оброблення результатів, а також зручність і зрозумілість користувацького інтерфейсу. Саме сукупність цих характеристик визначає ефективність використання систем перевірки знань у реальному освітньому середовищі та слугує основою для проєктування власного програмного продукту.

1.2. Система Moodle

Moodle є однією з найбільш поширених і відомих систем управління навчанням, яка активно використовується у закладах вищої, фахової передвищої та середньої освіти в усьому світі [1]. Дана платформа призначена для комплексної організації навчального процесу та забезпечує підтримку всіх його основних складових, зокрема створення і структурування навчальних курсів, розміщення теоретичних матеріалів, організацію практичних занять, проведення тестування та оцінювання результатів навчальної діяльності здобувачів освіти.

Однією з ключових переваг системи Moodle є гнучка та розвинена система ролей і прав доступу, яка дозволяє чітко розмежовувати функціональні повноваження між адміністраторами, викладачами та студентами. Це забезпечує контроль доступу до навчальних ресурсів і сприяє безпечному використанню системи. Moodle підтримує створення банків тестових завдань, їх категоризацію за темами та рівнем складності, випадкову генерацію запитань для кожної спроби проходження тесту, а також налаштування часових обмежень, кількості спроб і різних схем оцінювання результатів.

Крім того, платформа надає інструменти для збору статистики, аналізу успішності студентів і формування звітів, що дозволяє викладачам оперативно оцінювати рівень засвоєння навчального матеріалу та коригувати освітній процес. Завдяки підтримці додаткових модулів і плагінів Moodle може бути адаптована під специфічні потреби конкретного навчального закладу [2].

Разом з тим, використання Moodle має певні обмеження. Система потребує розгортання та налаштування серверного середовища, а також наявності кваліфікованих адміністративних ресурсів для її супроводу. Для початкових користувачів інтерфейс і логіка роботи платформи можуть здаватися складними та перевантаженими функціями, що збільшує час на навчання персоналу і впровадження системи в освітній процес. Це може ускладнювати швидкий початок роботи, особливо у випадках, коли необхідне оперативне розгортання системи контролю знань.

1.3. Google Forms

Google Forms є хмарним веб сервісом для створення анкет, опитувань і тестів, який широко використовується в освітній практиці завдяки простоті використання, доступності та інтеграції з іншими інструментами екосистеми Google. Сервіс не потребує встановлення додаткового програмного забезпечення, оскільки функціонує безпосередньо у веб браузері, що дозволяє використовувати його на різних пристроях та операційних системах. Це робить Google Forms зручним інструментом для швидкої організації перевірки знань у дистанційному та змішаному форматах навчання [3].

Однією з основних переваг Google Forms є висока швидкість створення тестів і опитувань. Викладач може за короткий проміжок часу підготувати форму, налаштувати типи запитань, визначити правильні відповіді та задати автоматичне оцінювання результатів. Додатковою перевагою є тісна інтеграція з іншими сервісами Google, зокрема Google Sheets, що дозволяє автоматично зберігати результати тестування у табличному вигляді, здійснювати базову статистичну обробку даних та будувати графіки й діаграми.

Завдяки простому інтерфейсу та мінімальному порогу входу Google Forms широко застосовується для проведення поточного, вхідного або діагностичного контролю знань. Платформа зручна для швидкого опитування студентів, збору зворотного зв’язку, перевірки засвоєння окремих тем або проведення коротких контрольних робіт. Результати тестування доступні практично миттєво, що дозволяє оперативно оцінювати рівень підготовки здобувачів освіти.

Разом з тим, Google Forms має низку обмежень, які знижують його придатність для використання як повноцінної системи комплексного контролю знань. Зокрема, сервіс не підтримує розвинену модель ролей користувачів, що ускладнює розмежування доступу між адміністраторами, викладачами та студентами. Також обмеженими є можливості створення складних сценаріїв тестування, таких як адаптивні тести, багаторівневе оцінювання або детальне налаштування параметрів проходження.

Аналітичні можливості Google Forms здебільшого зводяться до базової візуалізації результатів і потребують додаткової обробки даних у сторонніх інструментах. Відсутність вбудованих засобів глибокого аналізу успішності та контролю академічної доброчесності обмежує використання сервісу у великих навчальних закладах, де необхідні масштабовані та захищені рішення.

Таким чином, Google Forms є ефективним і зручним інструментом для швидкої організації опитувань і базового тестування, однак не може повністю задовольнити вимоги до системи комплексного контролю знань у масштабному освітньому середовищі. Це зумовлює доцільність розроблення спеціалізованого веб ресурсу, який поєднає простоту використання з розширеним функціоналом і аналітичними можливостями.

1.4. Kahoot!

Kahoot! є веб орієнтованою платформою для створення інтерактивних вікторин і тестів із використанням елементів гейміфікації, яка активно застосовується в освітньому процесі з метою підвищення зацікавленості та мотивації здобувачів освіти. Основна ідея платформи полягає у поєднанні перевірки знань із ігровими механіками, що дозволяє зробити навчання більш динамічним та привабливим для учнів і студентів.

Kahoot! надає можливість швидко створювати короткі тести та вікторини, які можуть проводитися як у синхронному режимі під час занять, так і в асинхронному форматі для самостійної роботи. Учасники проходять тестування з використанням персональних комп’ютерів або мобільних пристроїв, а результати відображаються в реальному часі. Такий підхід сприяє активній участі аудиторії та формує елемент змагання між учасниками.

Важливою особливістю Kahoot! є застосування гейміфікаційних елементів, зокрема системи балів, рейтингів, таймерів та візуальних ефектів. Це позитивно впливає на мотивацію здобувачів освіти, заохочує їх до швидкого реагування та повторення навчального матеріалу. Платформа особливо ефективна для

проведення вступних або підсумкових опитувань, актуалізації знань та закріплення вивчених тем у неформальній формі.

Разом з тим, функціональні можливості Kahoot! мають певні обмеження з точки зору академічного оцінювання. Платформа орієнтована переважно на короткі запитання з вибором відповіді та не підтримує складні сценарії тестування, що передбачають багаторівневе оцінювання, відкриті відповіді або детальну аналітику результатів. Це ускладнює використання Kahoot! для проведення серйозних контрольних або екзаменаційних робіт.

Аналітичні можливості системи обмежуються базовим переглядом результатів і загальної статистики, що не завжди є достатнім для глибокого аналізу успішності здобувачів освіти. Крім того, залежність від ігрових механік може знижувати об’єктивність оцінювання у випадках, коли важливу роль відіграє швидкість відповіді, а не глибина знань.

Таким чином, Kahoot! є ефективним інструментом для підвищення мотивації, активізації навчального процесу та проведення коротких перевірок знань у формі гри. Водночас платформа менш придатна для комплексного та формалізованого контролю знань у великих навчальних закладах, що підтверджує необхідність розроблення спеціалізованого веб ресурсу, орієнтованого на поєднання інтерактивності та повноцінного академічного оцінювання.

1.5. Classtime

Classtime є сучасною веборієнтованою платформою для організації перевірки знань та формувального оцінювання, яка активно використовується в освітньому процесі, зокрема під час очного, дистанційного та змішаного навчання. Основна концепція системи полягає у забезпеченні швидкого зворотного зв’язку між викладачем і здобувачами освіти, а також у візуалізації результатів навчальної діяльності в режимі реального часу.

Платформа Classtime надає викладачам можливість створювати різноманітні типи тестових завдань, включаючи запитання з вибором однієї або

кількох правильних відповідей, завдання на встановлення відповідності, числові відповіді та інші формати. Під час проходження тестування результати студентів відображаються у вигляді наочних таблиць і графіків, що дозволяє викладачеві оперативно оцінити загальний рівень засвоєння матеріалу та виявити проблемні теми.

Однією з ключових переваг Classtime є можливість використання платформи без складного налаштування та тривалого навчання персоналу. Інтерфейс системи є інтуїтивно зрозумілим, що дозволяє швидко інтегрувати її у навчальний процес. Платформа ефективно використовується для проведення поточного контролю знань, перевірки домашніх завдань, а також для формувального оцінювання під час занять.

Classtime підтримує роботу в режимі реального часу, що є особливо корисним для очних і онлайн-занять. Викладач може миттєво отримувати інформацію про відповіді студентів і коригувати подальший хід заняття залежно від рівня розуміння матеріалу. Такий підхід сприяє підвищенню ефективності навчального процесу та активнішій взаємодії між учасниками освітнього середовища [4].

Разом з тим, функціональні можливості Classtime значною мірою залежать від обраного тарифного плану. У безкоштовній версії платформи доступний обмежений набір інструментів, що може ускладнювати використання системи у великих навчальних групах або для проведення масштабного контролю знань. Крім того, за рівнем комплексності управління навчальним процесом Classtime поступається повноцінним системам управління навчанням, таким як Moodle.

Аналітичні можливості Classtime орієнтовані переважно на оперативний аналіз результатів, що є достатнім для формувального оцінювання, але може бути недостатнім для глибокого довгострокового аналізу успішності та ведення повноцінної статистики навчальних досягнень. Відсутність розширеної системи ролей і централізованого адміністрування також обмежує використання платформи як єдиного інструменту контролю знань у масштабі всього навчального закладу.

Таким чином, Classtime є ефективним засобом для оперативного контролю знань і підтримки інтерактивної взаємодії під час занять. Водночас обмеження функціональності та залежність від комерційних тарифів зумовлюють необхідність розроблення альтернативного веб ресурсу, який поєднуватиме зручність використання, гнучкість налаштувань і розширені аналітичні можливості для комплексного оцінювання результатів навчання.

1.6. Порівняльна таблиця та висновки

Для узагальнення результатів порівняння сформовано таблицю, яка відображає ключові можливості кожної системи за визначеними критеріями (табл. 1.1).

Таблиця 1.1 – Порівняння програмних систем перевірки знань

Критерій	Moodle	Google Forms	Kahoot!	Classtime
Тип рішення	LMS	Форми/опитування	Гейміфіковані вікторини	Онлайн-тестування + аналітика
Ролі користувачів	Так (розвинені)	Обмежено	Обмежено	Залежить від тарифу
Типи завдань	Широкий набір	Базові	Переважно короткі	Різні (у т.ч. формувальні)
Аналітика	Розвинена	Базова/через Sheets	Обмежена	Розвинена (реальний час)
Дистанційне навчання	Так	Так	Так	Так
Розгортання	Сервер/хмара	Хмара	Хмара	Хмара
Порог входу	Середній/високий	Низький	Низький	Низький/середній

У даному розділі було виконано порівняльний аналіз сучасних систем перевірки знань, які широко використовуються в освітньому процесі. Розглянуті платформи Moodle, Google Forms, Kahoot! та Classtime відрізняються за функціональними можливостями, складністю впровадження, рівнем автоматизації оцінювання та орієнтацією на різні сценарії використання.

Проведений аналіз показав, що система Moodle є найбільш комплексним і функціонально насиченим рішенням для організації навчання та контролю знань. Водночас її впровадження потребує значних технічних ресурсів і часу на налаштування. Google Forms, у свою чергу, забезпечує швидкий і зручний спосіб проведення базового тестування, проте має обмежені можливості щодо аналітики та керування користувачами.

Платформа Kahoot! ефективна для підвищення мотивації та залученості здобувачів освіти завдяки використанню гейміфікаційних елементів, однак менш придатна для формалізованого підсумкового оцінювання. Classtime забезпечує оперативний контроль знань і зручну візуалізацію результатів, але залежність від тарифних планів і обмежений функціонал безкоштовної версії звужують сферу її застосування.

Отже, жодна з розглянутих систем не повністю задовольняє всі вимоги сучасного освітнього середовища щодо автоматизації тестування, гнучкості налаштувань, аналітичних можливостей та простоти використання. Це обґрунтовує доцільність розроблення власного веб ресурсу, який поєднає переваги наявних рішень і мінімізує їхні недоліки.

Результати порівняльного аналізу можуть бути використані як основа для формування функціональних і технічних вимог до проєктованої системи контролю знань, а також для подальшого проєктування архітектури та реалізації веб ресурсу, описаних у наступних розділах роботи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБ РЕСУРСУ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ

2.1 Обґрунтування технологій і засобів вирішення поставленого завдання

Для реалізації поставленого завдання з розроблення веб ресурсу для організації та проведення тестування й опитувань у навчальному процесі було обрано набір сучасних, поширених і перевірених технологій веб розроблення. Вибір програмних засобів здійснювався з урахуванням вимог до функціональності системи, масштабованості, безпеки, зручності користування, а також можливості подальшого розширення й супроводу програмного продукту.

Основною вимогою до розроблюваного веб ресурсу є забезпечення динамічної взаємодії з користувачами, автоматизація процесів тестування та збереження результатів у структурованому вигляді. У зв’язку з цим доцільним є використання клієнт-серверної архітектури, яка дозволяє розділити оброблення даних, бізнес-логіку та представлення інформації.

Для реалізації клієнтської частини веб ресурсу було обрано мову програмування JavaScript, яка є стандартом для створення інтерактивних веб інтерфейсів [5]. Використання JavaScript дозволяє реалізувати динамічну зміну вмісту сторінок без їх повного перезавантаження, що значно підвищує зручність роботи користувача та швидкість взаємодії із системою. Крім того, JavaScript забезпечує можливість перевірки коректності введених даних на стороні клієнта та реалізації інтерактивних елементів інтерфейсу [6].

Для серверної частини веб ресурсу було обрано мову програмування PHP, яка широко застосовується для розроблення веб орієнтованих інформаційних систем [7]. PHP забезпечує ефективну обробку HTTP-запитів, реалізацію бізнес-логіки, керування користувацькими сесіями та взаємодію з базою даних. Перевагами PHP є простота використання, велика кількість готових бібліотек, висока продуктивність у веб середовищі та сумісність з більшістю веб серверів.

Для зберігання та оброблення даних було обрано систему керування базами даних MySQL, яка забезпечує надійне збереження інформації про

користувачів, тести, запитання та результати тестування. MySQL підтримує реляційну модель даних, що дозволяє логічно структурувати інформацію та забезпечити її цілісність за допомогою механізмів первинних і зовнішніх ключів. Використання MySQL є доцільним з огляду на її продуктивність, масштабованість і широку підтримку у веб проєктах [8].

Для забезпечення взаємодії між клієнтською та серверною частинами системи застосовується технологія AJAX (Asynchronous JavaScript and XML). Використання AJAX дозволяє передавати дані між браузером і сервером у фоновому режимі без перезавантаження веб сторінки. Це забезпечує більш плавну роботу веб ресурсу, оперативне відображення результатів тестування та підвищує загальну зручність користування системою.

Як веб сервер для розгортання розроблюваного ресурсу обрано Apache, який є одним із найпоширеніших серверних рішень у веб розробленні [9]. Apache забезпечує стабільну роботу веб додатків, підтримує модульну архітектуру та має тісну інтеграцію з мовою PHP і системою керування базами даних MySQL. Це робить його доцільним вибором для реалізації серверної інфраструктури веб ресурсу.

Загалом обраний стек технологій JavaScript – PHP – MySQL – Apache – AJAX є оптимальним для розв’язання поставленого завдання, оскільки поєднує у собі доступність, функціональність, масштабованість і простоту впровадження. Використання зазначених засобів дозволяє створити ефективний, надійний і зручний вебресурс для контролю знань, який може бути інтегрований у сучасне освітнє середовище та адаптований до потреб дистанційного навчання.

2.2 Загальна архітектура веб ресурсу

Архітектура веб ресурсу визначає загальні принципи побудови програмної системи, взаємодію її компонентів та спосіб оброблення інформації. Від правильно обраної архітектури значною мірою залежить ефективність, масштабованість, надійність і зручність супроводу веб-ресурсу. У межах даної роботи для реалізації системи організації та проведення тестування обрано

класичну клієнт-серверну архітектуру з трьох ланковою структурою, яка є загальноприйнятою для сучасних веб-орієнтованих інформаційних систем (рис. 2.1).

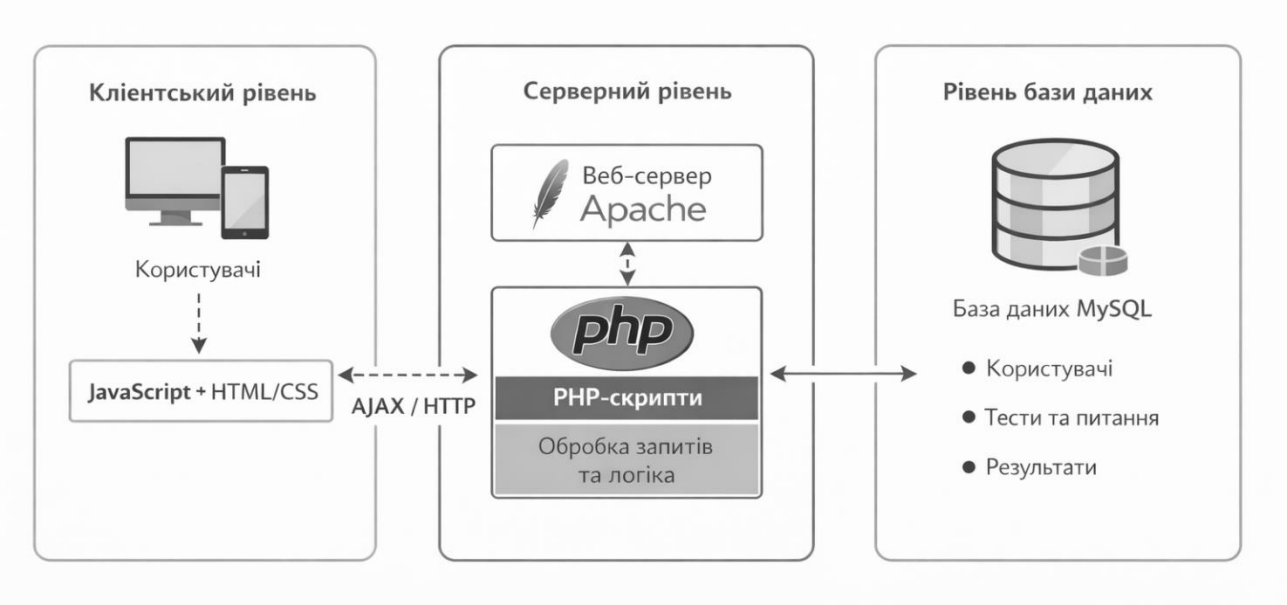

Рисунок 2.1 – Три ланкова архітектура програмного комплексу

Трирівнева архітектура передбачає поділ системи на клієнтський рівень, серверний рівень та рівень бази даних. Такий підхід забезпечує логічне розмежування функціональних обов’язків між компонентами системи, що спрощує її розроблення, тестування та подальший розвиток.

Клієнтський рівень відповідає за взаємодію користувача з веб ресурсом і реалізується у вигляді веб інтерфейсу, доступного через стандартний веб браузер. На цьому рівні здійснюється відображення інформації, введення користувацьких даних, навігація між сторінками та первинна перевірка введених значень.

Для реалізації клієнтського рівня використовується мова програмування JavaScript, а також стандартні веб технології HTML і CSS. JavaScript забезпечує динамічну зміну вмісту веб сторінок, реалізацію інтерактивних елементів інтерфейсу та асинхронну взаємодію із сервером. Завдяки цьому користувач отримує швидкий відгук системи без необхідності повного перезавантаження сторінок.

Клієнтський рівень підтримує роботу різних категорій користувачів – адміністратора, викладача та студента. Інтерфейс адаптується відповідно до ролі користувача, надаючи доступ лише до дозволених функцій. Такий підхід підвищує зручність використання системи та зменшує ризик помилкових дій.

Серверний рівень є центральним компонентом архітектури веб ресурсу та відповідає за реалізацію бізнес-логіки системи. На цьому рівні здійснюється оброблення HTTP-запитів, автентифікація та авторизація користувачів, керування тестами й опитуваннями, а також формування відповідей для клієнтського рівня.

Для реалізації серверної логіки використовується мова програмування PHP, яка забезпечує ефективну взаємодію з веб сервером та базою даних. Серверний рівень обробляє дані, отримані від клієнта, перевіряє їх коректність, виконує необхідні обчислення та повертає результати у зручному для відображення форматі.

Окрему увагу приділено реалізації механізмів керування сесіями користувачів, що дозволяє зберігати стан роботи користувача під час взаємодії з веб ресурсом. Серверний рівень також відповідає за контроль доступу до функцій системи відповідно до ролей користувачів, що забезпечує безпеку та цілісність даних.

Рівень бази даних відповідає за збереження, оброблення та структуровану організацію інформації, необхідної для функціонування веб ресурсу. Для реалізації цього рівня використовується система керування базами даних MySQL, яка підтримує реляційну модель даних та забезпечує високу продуктивність при роботі з великими обсягами інформації.

У базі даних зберігаються відомості про користувачів, їх ролі, тести, запитання, варіанти відповідей та результати проходження тестування. Структура бази даних проєктується з урахуванням принципів нормалізації, що дозволяє уникнути дублювання даних і забезпечити їх логічну цілісність. Взаємодія між таблицями реалізується за допомогою первинних і зовнішніх ключів.

Взаємодія між рівнями. Обмін даними між клієнтським і серверним рівнями здійснюється за допомогою HTTP-запитів, а для підвищення швидкодії та зручності користування застосовується технологія AJAX. Це дозволяє передавати дані у фоновому режимі та оновлювати окремі частини веб сторінки без повного перезавантаження.

Така організація взаємодії між компонентами системи забезпечує швидку реакцію веб ресурсу на дії користувача, зменшує навантаження на сервер і покращує загальне враження від роботи з системою.

Обрана архітектура веб ресурсу має низку переваг, зокрема:

- чіткий розподіл функцій між компонентами системи;
- можливість масштабування та розширення функціоналу;
- зручність супроводу та модернізації;
- підвищений рівень безпеки та надійності;
- адаптованість до використання в умовах дистанційного навчання.

Таким чином, клієнт-серверна трирівнева архітектура є доцільним і обґрунтованим вибором для реалізації веб ресурсу з організації та проведення тестування й опитувань, оскільки вона забезпечує ефективну взаємодію користувачів із системою та створює основу для подальшого розвитку програмного продукту.

2.3 Серверна частина веб-ресурсу

Серверна частина веб-ресурсу є центральним елементом його архітектури та виконує ключову роль у забезпеченні функціонування всієї системи. Саме на серверному рівні реалізується бізнес-логіка веб ресурсу, здійснюється оброблення запитів користувачів, керування доступом до функціональних можливостей системи, а також взаємодія з базою даних.

Архітектура серверної частини побудована на основі клієнт-серверної моделі з використанням веб сервера Apache та мови програмування PHP (рис. 2.2). Такий підхід є традиційним і водночас ефективним для створення веб

орієнтованих інформаційних систем, оскільки забезпечує стабільність, масштабованість і зручність супроводу програмного продукту.

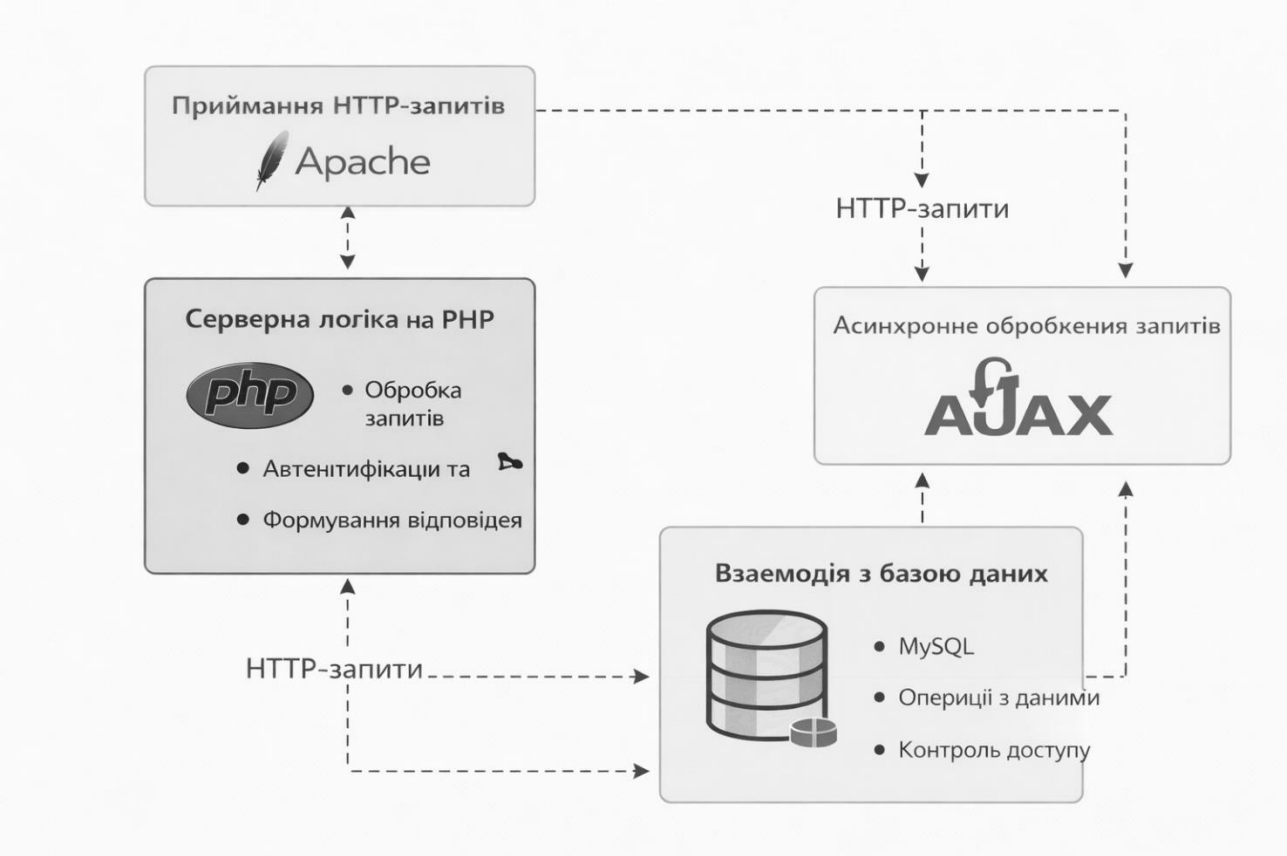

Рисунок 2.2 – Структура серверної частини програмного комплексу

Веб сервер Apache відповідає за приймання HTTP-запитів від клієнтської частини системи та передачу їх на оброблення серверним скриптам. Apache забезпечує коректну обробку запитів, керування з’єднаннями, а також підтримує механізми безпеки, зокрема налаштування доступу до ресурсів і захист від несанкціонованого доступу.

Завдяки модульній архітектурі Apache може бути легко розширений додатковими модулями, що дозволяє адаптувати сервер до потреб конкретного веб ресурсу. Тісна інтеграція Apache з мовою PHP забезпечує ефективну обробку динамічного контенту та стабільну роботу серверної частини системи.

Основна бізнес-логіка веб ресурсу реалізується за допомогою мови програмування PHP. PHP-скрипти виконують оброблення даних, отриманих від клієнтського рівня, перевірку коректності введеної інформації, а також виконують операції створення, редагування та видалення тестів і опитувань.

Серверна частина відповідає за автентифікацію та авторизацію користувачів, що дозволяє розмежовувати доступ до функціональних можливостей системи відповідно до ролей (адміністратор, викладач, студент). Такий механізм підвищує безпеку системи та запобігає несанкціонованому доступу до даних.

PHP-скрипти також відповідають за формування відповідей серверу, які передаються клієнтській частині у структурованому вигляді. Це дозволяє забезпечити коректне відображення результатів тестування, статистичних даних і повідомлень користувачеві.

Серверна частина здійснює безпосередню взаємодію з базою даних MySQL, у якій зберігається інформація про користувачів, тести, запитання та результати тестування. Доступ до бази даних реалізується через відповідні інтерфейси, що дозволяє виконувати операції вибірки, додавання, оновлення та видалення даних.

Усі операції з базою даних виконуються на серверному рівні, що виключає прямий доступ клієнта до даних і підвищує рівень безпеки системи. Такий підхід також дозволяє забезпечити цілісність інформації та контроль коректності виконання запитів.

Для підвищення швидкодії та зручності користування веб ресурсом серверна архітектура підтримує асинхронну обробку запитів за допомогою технології AJAX. У цьому випадку клієнтська частина надсилає запити до серверу без повного перезавантаження сторінки, а сервер повертає лише необхідні дані.

Асинхронна взаємодія дозволяє оперативно відображати результати тестування, оновлювати інформацію та забезпечувати більш плавну роботу користувацького інтерфейсу.

Запропонована архітектура серверної частини має низку переваг:

- централізована реалізація бізнес-логіки;
- підвищений рівень безпеки та контролю доступу;
- можливість масштабування та оптимізації продуктивності;
- зручність супроводу та модернізації програмного забезпечення;

- сумісність з поширеними веб технологіями.

Таким чином, архітектура серверної частини веб ресурсу є логічно обґрунтованою та відповідає вимогам сучасних веборієнтованих інформаційних систем. Вона забезпечує надійну роботу веб ресурсу, ефективну обробку запитів користувачів і створює основу для подальшого розвитку та розширення функціональних можливостей системи.

2.4 Проєктування клієнтської частини програми

Для побудови клієнтської частини веб ресурсу доцільно застосувати архітектурний шаблон MVVM (Model–View–ViewModel), який забезпечує чітке розмежування відповідальностей між відображенням даних, прикладною логікою та моделлю даних (рис. 2.3). Використання MVVM підвищує підтримуваність коду, спрощує тестування та дозволяє масштабувати інтерфейс без суттєвого ускладнення проєкту [11].

MVVM передбачає поділ клієнтського застосунку на три логічні компоненти:

1. Model (Модель) – описує структури даних та правила роботи з ними (предметна область).
2. View (Подання) – відповідає за відображення інтерфейсу та взаємодію з користувачем.
3. ViewModel (Модель подання) – проміжна ланка між View і Model; містить логіку підготовки даних для відображення, керує станом інтерфейсу та обробляє дії користувача.

Ключовою характеристикою MVVM є двостороннє зв’язування даних (data binding) або його еквівалент через реактивний стан: зміни у ViewModel автоматично відображаються у View, а події у View ініціюють виклики методів ViewModel.

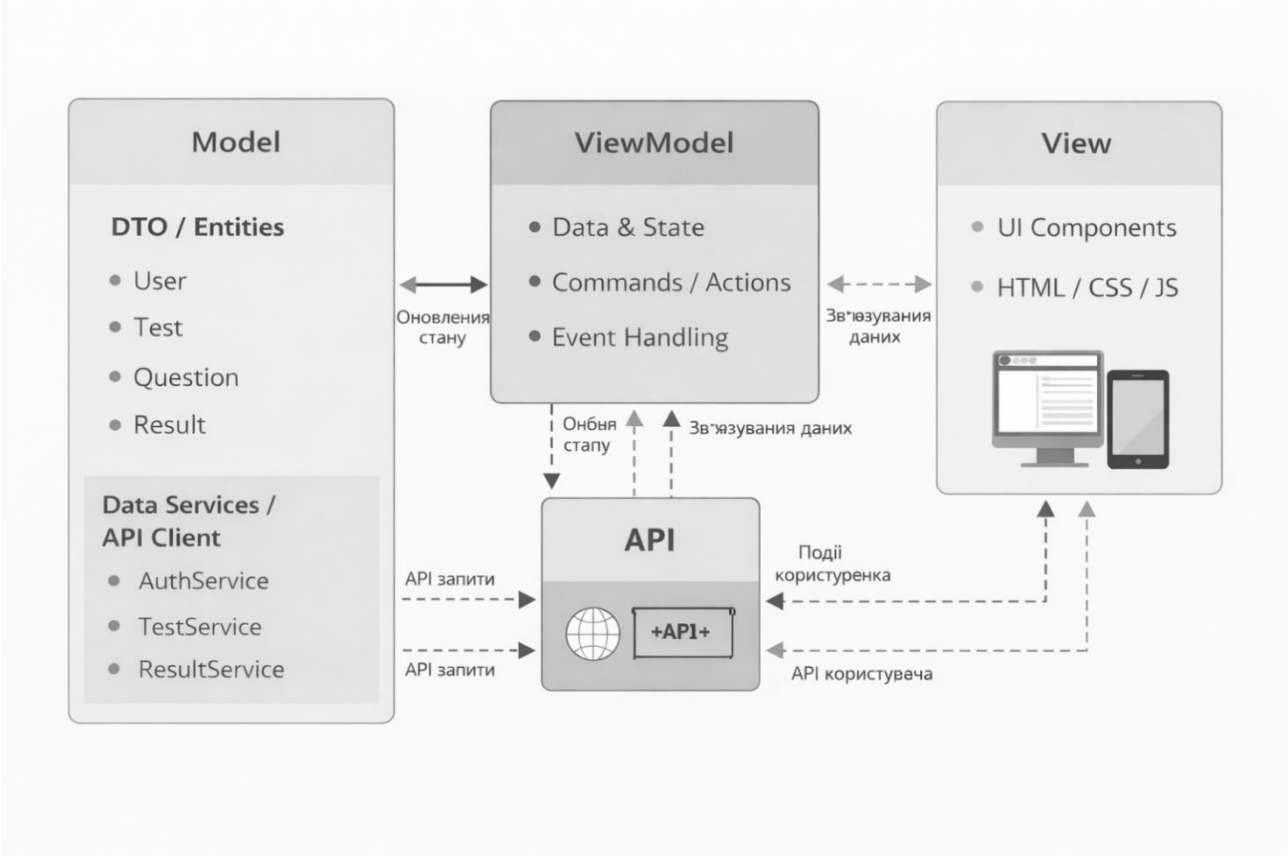

Рисунок 2.3 – Структура клієнтської частини програмного комплексу за шаблоном MVVM

Model у клієнтській частині системи включає:

1. Доменно-логічні сутності (DTO/Entities):
   - User (користувач: id, роль, ПІБ/логін, статус);
   - Test (тест: id, назва, автор, параметри доступу, час, спроби);
   - Question (питання: id, testId, тип, текст, бали);
   - Option/Answer (варіанти/відповідь: id, questionId, текст, ознака правильності);
   - Attempt/Result (спроба/результат: userId, testId, дата, бал, деталізація).
2. Сервіси доступу до даних (Data Services / API Client) Модуль, що реалізує запити до серверної частини (REST/HTTP), наприклад:
   - AuthService (вхід, вихід, токен/сесія);
   - TestService (отримання списку тестів, створення/редагування);

- AttemptService (проходження тесту, збереження відповідей, отримання результатів).

3. Механізми кешування/сховища стану (за потреби) Наприклад, збереження токена, даних профілю, останніх переглянутих тестів.

Model не повинен містити логіку відображення або залежати від UI. Його призначення – надати ViewModel узгоджені структури й операції роботи з даними.

View – це інтерфейс користувача, реалізований у вигляді вебсторінок/компонентів (HTML/CSS/JS). View виконує такі функції:

- відображає дані зі ViewModel (назви тестів, запитання, таймер, результати тощо);
- приймає введення користувача (вибір відповідей, натискання кнопок, введення логіну/пароля);
- ініціює події (наприклад, onLoginClick, onStartTest, onSubmitAnswer);
- не містить бізнес-логіки; допускається лише мінімальна логіка представлення (валідація формату, відображення помилок, маски вводу).

Приклади View у WebMentor:

- сторінка входу;
- панель викладача (створення тестів/питань);
- сторінка проходження тесту;
- сторінка результатів;
- сторінка адміністрування користувачів.

ViewModel є ключовим компонентом MVVM. Він:

- зберігає стан UI (поточний користувач, список тестів, активний тест, прогрес проходження, помилки валідації, завантаження/очікування);
- надає View дані у форматі, зручному для відображення (наприклад, відфільтровані списки, агреговані підсумки);
- реалізує команди/дії (commands), які викликаються з View: login(), logout(), loadTests(), createTest(), addQuestion(), startAttempt(), submitAnswer(), finishAttempt(), loadResults().

ViewModel взаємодіє з Model через API/сервіси та оновлює стан. У MVVM ViewModel не повинен напряму маніпулювати DOM; його роль – підготувати дані і реактивно повідомити View про зміни.

Типовий сценарій взаємодії в MVVM:

1. Користувач виконує дію у View (наприклад, натискає «Увійти»).
2. View викликає відповідну команду ViewModel (login()), передаючи введені дані.
3. ViewModel звертається до Model/API (AuthService.login()), обробляє відповідь.
4. ViewModel оновлює власний стан (user, token, error).
5. View автоматично оновлюється через data binding/реактивні механізми, відображаючи результат (перехід, повідомлення, оновлення UI).

Такий підхід мінімізує зв’язність між UI та логікою, робить систему передбачуваною й зручною для тестування.

Застосування MVVM у клієнтській частині Web Mentor забезпечує:

- розділення відповідальностей між інтерфейсом і логікою;
- спрощене модульне тестування (особливо ViewModel і Model);
- масштабованість (легко додавати нові екрани та сценарії);
- підтримуваність коду (менше дублювання, чітка структура);
- стабільність UI при розширенні функціоналу (View залишається “тонким”);
- зручність роботи з асинхронними запитами (стани loading/error/success централізовано у ViewModel).

Приклад логічного розподілу модулів у MVVM:

Model:

- models/ (User, Test, Question, Attempt);
- services/ (AuthService, TestService, ResultService);
- storage/ (tokenStorage, cache);
- ViewModel;
- viewmodels/AuthVM (вхід/вихід, стани авторизації);
- viewmodels/TeacherVM (керування тестами/питаннями);

- viewmodels/StudentVM (проходження тестів, відповіді);
- viewmodels/AdminVM (користувачі, ролі).

View:

- views/LoginView;
- views/TeacherDashboardView;
- views/TestRunnerView;
- views/ResultsView;
- views/AdminPanelView.

2.5 Проєктування алгоритму роботи програми

Проєкт алгоритму відображає логічну та послідовну структуру основних етапів роботи веб ресурсу Web Mentor, починаючи з моменту входу користувача до системи та завершуючи закінченням сеансу роботи. Алгоритм описує порядок взаємодії користувача з програмною системою, а також механізми оброблення запитів, що виникають у процесі використання функціональних можливостей веб ресурсу.

Логіка роботи системи ґрунтується на принципі автентифікації та подальшої перевірки ролі користувача, що дозволяє коректно і безпечно організувати доступ до різних модулів системи. Після успішного входу веб ресурс автоматично визначає тип користувача (адміністратор, викладач або студент) і надає доступ лише до тих функцій, які передбачені відповідним рівнем доступу. Такий підхід забезпечує чітке розмежування повноважень та запобігає несанкціонованому використанню функціоналу системи.

Кожен етап алгоритму передбачає виконання конкретних дій, пов’язаних з обробленням даних, відображенням інформації та збереженням результатів у базі даних (рис. 2.4). У процесі роботи користувач може виконувати дозволені операції, після чого система коректно завершує сеанс, забезпечуючи цілісність даних та стабільність функціонування веб ресурсу. Таким чином, проєкт алгоритму дозволяє наочно представити загальну логіку роботи Web Mentor і слугує основою для подальшої програмної реалізації системи.

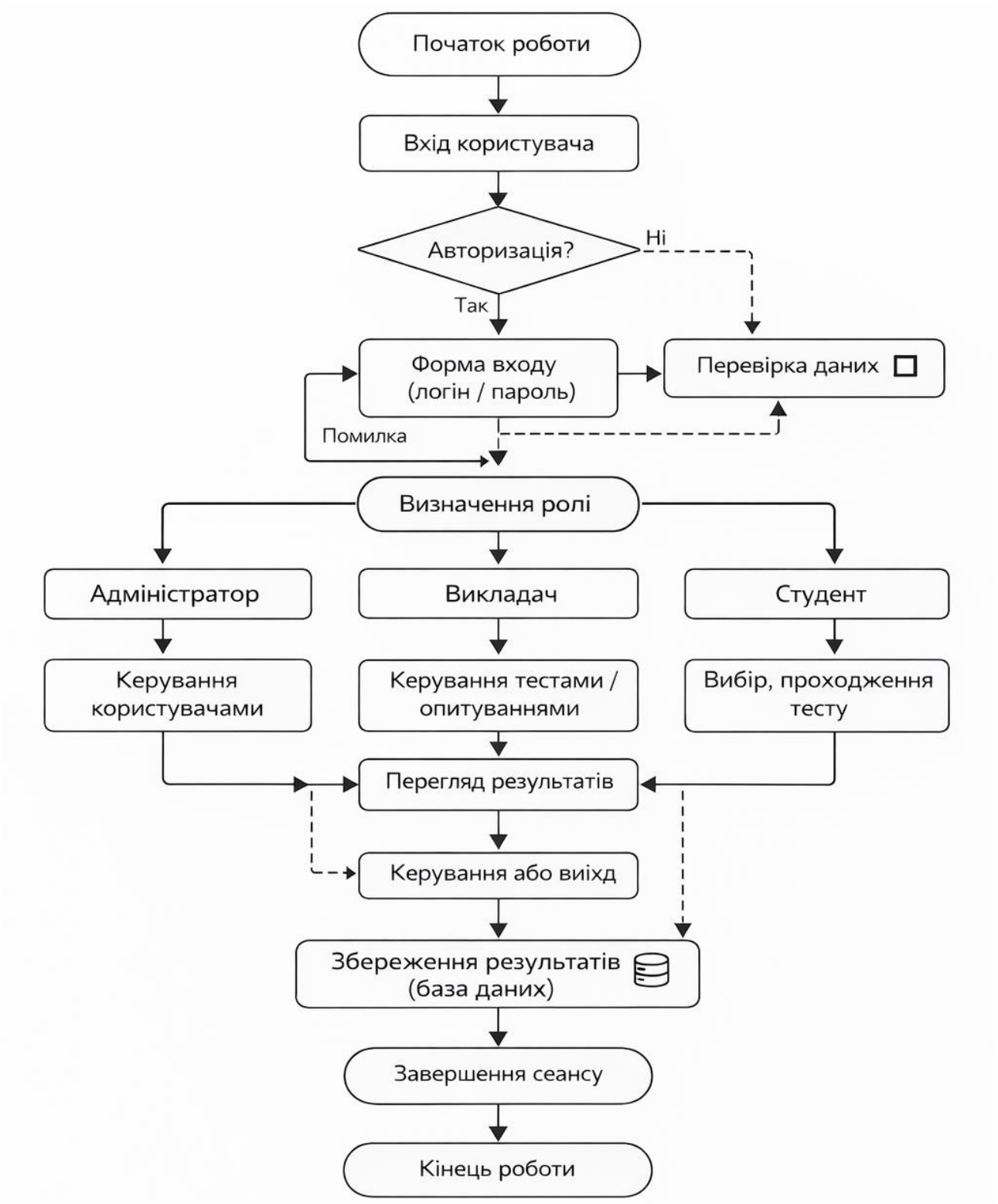

Рисунок 2.4 – Блок-схема роботи програми

1. Початок роботи системи. Користувач відкриває веб ресурс у браузері. Клієнтська частина ініціює завантаження інтерфейсу та перевіряє наявність активної сесії/токена авторизації.

2. Перевірка автентифікації. Якщо активна сесія відсутня, система відображає форму входу. Користувач вводить облікові дані (логін/пароль), після чого клієнт надсилає запит на сервер для перевірки даних.

3. Обробка даних входу на сервері. Сервер виконує валідацію введених даних, порівнює їх із записами в базі даних та визначає роль користувача. Якщо дані некоректні – формується повідомлення про помилку, і користувач повертається до форми входу. Якщо дані коректні – створюється сесія/токен, користувач переходить до головної сторінки.

4. Визначення ролі користувача. Після успішної авторизації система виконує розгалуження за роллю: Адміністратор / Викладач / Студент. Кожна роль отримує доступ лише до передбачених модулів.

5. Роль “Адміністратор”. Адміністратор отримує доступ до модуля керування користувачами:

- створення/реєстрація облікових записів;
- редагування даних користувачів;
- деактивація або видалення застарілих записів;
- перегляд службової інформації (за потреби).

Після виконання дій адміністратор може повернутися до панелі керування або завершити сеанс (вихід).

6. Роль “Викладач”. Викладач переходить до модуля керування тестами та опитуваннями:

- створення тесту/опитування, налаштування параметрів (час, кількість спроб, доступність);
- формування структури тесту: додавання запитань, варіантів відповідей, балів;
- публікація тесту для студентів (активація доступу).

Окремою гілкою є перегляд результатів:

- вибір тесту;
- перегляд списку спроб студентів;
- перегляд підсумкових балів і деталізації відповідей.

Після завершення викладач повертається до панелі або виходить із системи.

7. Роль “Студент”. Студент отримує доступ до списку доступних тестів/опитувань і може:
   - виконати пошук потрібного тесту;
   - відкрити тест і розпочати проходження;
   - відповідати на запитання (з покроковим збереженням відповідей або збереженням при завершенні – залежно від налаштувань).

Після натискання “Завершити” система виконує:
   - перевірку відповідей (автоматичне оцінювання);
   - формування підсумкового результату;
   - збереження результату в базі даних;
   - відображення результатів студенту (бал, помилки, правильні відповіді — якщо дозволено правилами тесту).

8. Збереження та обробка результатів. Усі результати тестування зберігаються в базі даних. Сервер забезпечує цілісність даних та прив’язку результатів до конкретного користувача, тесту і часу проходження. На цьому етапі також можуть виконуватися додаткові операції: формування статистики, підрахунок середніх значень, підготовка звітів.
9. Завершення сеансу. Користувач за потреби виконує вихід із системи. Сервер завершує сесію/анулює токен, після чого клієнт повертається на сторінку входу.
10. Кінець роботи. Робота програми завершується або повторюється для нового користувача/нового сеансу.

Проєкт алгоритму роботи веб ресурсу Web Mentor наочно відображає загальну логіку функціонування програмної системи. Запропонований проєкт алгоритму дозволяє послідовно простежити основні етапи взаємодії користувача з веб ресурсом, починаючи з процесу автентифікації та завершуючи коректним завершенням сеансу роботи. Проєкт алгоритму також демонструє взаємозв’язок між клієнтською та серверною частинами системи, процеси оброблення запитів і збереження результатів у базі даних. Це дозволяє використовувати його як основу для подальшої програмної реалізації, тестування та вдосконалення веб ресурсу.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОБОТИ ВЕБ РЕСУРСУ

3.1 Обґрунтування та вибір середовищ та фреймворків для розробки веб ресурсу

Під час розроблення веб ресурсу Web Mentor було використано комплекс сучасних програмних інструментів, які забезпечують ефективну реалізацію клієнтської та серверної частин системи, а також зручну роботу з базою даних. Вибір середовищ розробки та фреймворків здійснювався з урахуванням вимог до продуктивності, зручності налагодження, масштабованості та якості програмного коду.

Для розроблення клієнтської частини веб ресурсу було обрано інтегроване середовище розробки WebStorm у поєднанні з фреймворком Vue.js. WebStorm є професійним IDE для JavaScript-розробки, яке забезпечує потужні засоби редагування коду, інтелектуальне автодоповнення, аналіз помилок у реальному часі та зручні інструменти налагодження. Фреймворк Vue.js було обрано завдяки його простоті, гнучкості та підтримці архітектурного шаблону MVVM, що повністю відповідає вимогам до клієнтської частини Web Mentor. Vue.js дозволяє створювати реактивні інтерфейси, забезпечує двостороннє зв’язування даних і спрощує керування станом інтерфейсу. Завдяки компонентному підходу Vue.js полегшує повторне використання коду та спрощує масштабування клієнтської частини веб ресурсу. Поєднання WebStorm і Vue.js є оптимальним вибором для створення сучасного, зручного та швидкодіючого інтерфейсу користувача.

Для реалізації серверної логіки веб ресурсу Web Mentor було використано інтегроване середовище розробки PHPStorm, яке є одним із найпотужніших інструментів для PHP-розробки. PHPStorm забезпечує повноцінну підтримку мови PHP, зокрема автодоповнення, статичний аналіз коду, підсвічування помилок і зручні засоби налагодження. Використання PHPStorm дозволяє ефективно реалізовувати бізнес-логіку веб ресурсу, керувати сесіями користувачів, обробляти запити та інтегрувати серверну частину з базою даних.

IDE підтримує роботу з популярними шаблонами проєктування та сприяє написанню структурованого й читабельного коду, що є важливим для довготривалого супроводу проєкту. Таким чином, PHPStorm є доцільним вибором для створення стабільної та надійної серверної частини Web Mentor.

Для проєктування, адміністрування та керування базою даних веб ресурсу Web Mentor використовується MySQL Workbench. Це офіційний інструмент для роботи з MySQL, який надає засоби візуального проєктування структури бази даних, створення ER-діаграм, редагування таблиць і виконання SQL-запитів. MySQL Workbench дозволяє ефективно проєктувати реляційну структуру даних, аналізувати зв'язки між таблицями та забезпечувати цілісність інформації. Завдяки графічному інтерфейсу спрощується процес адміністрування бази даних і підвищується наочність структури збережених даних. Використання MySQL Workbench є оптимальним рішенням для роботи з базою даних Web Mentor, оскільки інструмент забезпечує зручність, надійність і відповідність сучасним вимогам до проєктування даних.

Для розгортання та тестування веб ресурсу WebMentor у локальному середовищі було використано програмний комплекс XAMPP, який включає веб сервер Apache, інтерпретатор PHP та сервер бази даних MySQL. XAMPP дозволяє швидко налаштувати повноцінне серверне середовище без складних конфігурацій. Використання XAMPP забезпечує можливість тестування роботи веб ресурсу в умовах, максимально наближених до реального серверного середовища. Це дозволяє виявляти та усувати помилки на ранніх етапах розроблення, перевіряти взаємодію клієнтської та серверної частин і коректність роботи з базою даних. Завдяки простоті встановлення та стабільній роботі XAMPP є доцільним і ефективним вибором для розроблення та налагодження веб ресурсу Web Mentor.

Обраний набір інструментальних засобів – WebStorm і Vue.js для JavaScript, PHPStorm для PHP, MySQL Workbench для роботи з базою даних та XAMPP для локального серверного середовища – є оптимальним для реалізації веб ресурсу Web Mentor. Ці засоби забезпечують високу якість програмного

коду, зручність розроблення, масштабованість і надійність системи, що повністю відповідає вимогам сучасного освітнього веб ресурсу.

3.2 Реалізація бази даних для веб ресурсу

База даних веб ресурсу Web Mentor призначена для зберігання, оброблення та забезпечення цілісності інформації, необхідної для функціонування системи організації тестування та опитувань у навчальному процесі (рис. 3.1).

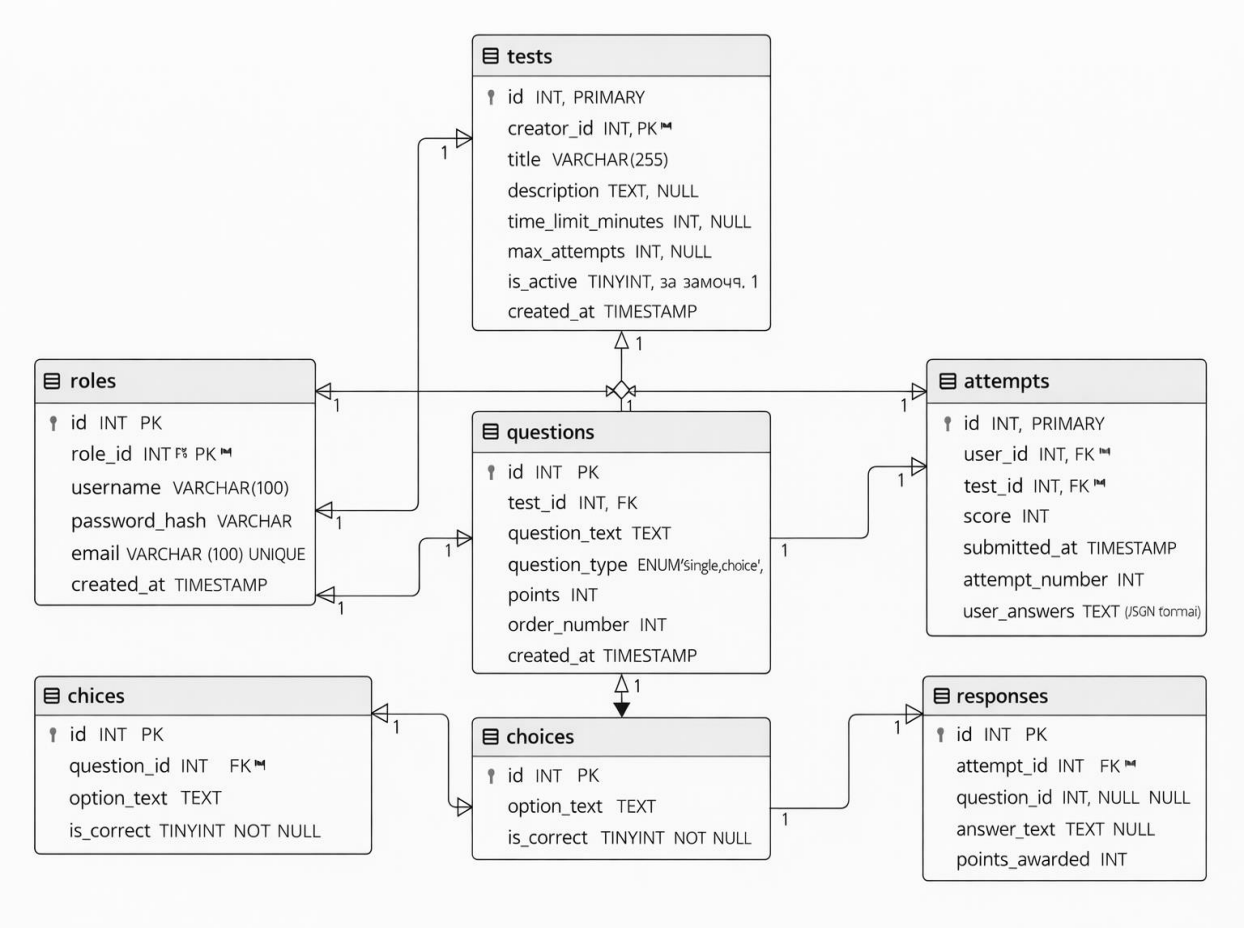

Рисунок 3.1 – Схеми об’єктів бази даних

Структура бази даних побудована за реляційною моделлю та розроблена з урахуванням вимог до масштабованості, безпеки та зручності оброблення даних.

Основними об’єктами предметної області є користувачі, тести, запитання, варіанти відповідей, спроби проходження тестів, результати, а також елементи

опитувань. Для кожної сутності створено окрему таблицю, між якими реалізовано логічні зв’язки типу «один-до-багатьох».

Таблиця roles використовується для зберігання переліку ролей користувачів системи. Кожен запис містить унікальний ідентифікатор ролі та її назву. Дана таблиця дозволяє реалізувати гнучку систему розмежування доступу до функціональних можливостей веб ресурсу.

Таблиця users зберігає інформацію про користувачів системи. Вона містить дані, необхідні для автентифікації та ідентифікації користувачів, зокрема унікальний ідентифікатор, посилання на роль користувача, персональні дані, облікову інформацію та статус облікового запису. Зв’язок із таблицею roles забезпечує коректне визначення прав доступу.

Таблиця tests призначена для зберігання інформації про тести, створені викладачами. Вона містить назву тесту, опис, параметри проходження (часові обмеження, кількість спроб), а також ознаку публікації. Кожен тест пов’язаний із конкретним викладачем через зовнішній ключ, що посилається на таблицю users.

Таблиця questions містить перелік запитань, що входять до складу тестів. Для кожного запитання зберігається текст, тип запитання, кількість балів та порядок відображення. Таблиця пов’язана з таблицею tests зв’язком «один-до-багатьох», оскільки один тест може містити декілька запитань.

Таблиця options призначена для зберігання варіантів відповідей на тестові запитання. Кожен варіант відповіді пов’язаний з конкретним запитанням і містить ознаку правильності. Така структура дозволяє реалізувати автоматизоване оцінювання відповідей студентів.

Таблиця attempts зберігає інформацію про спроби проходження тестів студентами. Вона містить дані про тест, користувача, час початку та завершення проходження, підсумковий бал і статус спроби. Зв’язки з таблицями tests та users забезпечують коректну ідентифікацію спроби.

Таблиця responses призначена для збереження відповідей студентів на окремі запитання під час проходження тестів. Вона містить посилання на спробу тестування, запитання та, за необхідності, обраний варіант відповіді або

введений текст. Таблиця також зберігає інформацію про правильність відповіді та кількість нарахованих балів.

Таблиця polls використовується для зберігання інформації про опитування, створені викладачами. Вона містить назву, опис, ознаку активності та дату створення опитування. Кожне опитування пов’язане з користувачем-автором.

Таблиця poll_questions містить перелік запитань, що входять до складу опитувань. Вона забезпечує можливість створення структурованих анкет із декількох запитань та визначення їх порядку.

Таблиця poll_answers призначена для зберігання відповідей студентів на запитання опитувань. Кожен запис містить посилання на запитання опитування, користувача та введену відповідь, а також дату її створення.

Схема бази даних WebMentor побудована з дотриманням принципів нормалізації, що забезпечує мінімізацію надлишковості даних і підвищує цілісність інформації. Використання первинних і зовнішніх ключів дозволяє підтримувати логічні зв’язки між таблицями та гарантує коректність збережених даних. Запропонована структура є гнучкою та придатною для розширення функціональних можливостей вебресурсу в майбутньому.

3.3 Реалізація програмного коду для клієнтської частини веб ресурсу

Екран входу (Login) входу забезпечує:

- введення email/пароля;
- клієнтську валідацію;
- відправлення запиту на сервер (/api/auth/login);
- збереження токена/ролі;
- перенаправлення користувача на відповідне головне вікно (студент/викладач).

Склад програмного модулю у термінах MVVM:

- View: форма входу (LoginView.vue)
- ViewModel: реактивний стан form, loading, error + метод submit()
- Model/API: authApi.login()

Текст програмного модулю src/views/LoginView.vue :

```
<template>
 <section class="page">
  <div class="card">
   <h1>Вхід до WebMentor</h1>

   <form @submit.prevent="submit">
    <label>
     Email
     <input v-model.trim="form.email" type="email" autocomplete="username" />
    </label>

    <label>
     Пароль
     <input v-model="form.password" type="password" autocomplete="current-password" />
    </label>

    <p v-if="error" class="error">{{ error }}</p>

    <button :disabled="loading">
     {{ loading ? "Вхід..." : "Увійти" }}
    </button>
   </form>
  </div>
 </section>
</template>

<script setup>
import { reactive, ref } from "vue";
import { useRouter } from "vue-router";
import { authApi } from "../api/authApi";
import { useAuthStore } from "../stores/authStore";

const router = useRouter();
const auth = useAuthStore();

const form = reactive({ email: "", password: "" });
const loading = ref(false);
const error = ref("");

function validate() {
 if (!form.email || !form.password) return "Заповніть email і пароль.";
 if (!form.email.includes("@")) return "Некоректний формат email.";
 if (form.password.length < 6) return "Пароль має містити щонайменше 6 символів.";
 return "";
}

async function submit() {
 error.value = validate();
 if (error.value) return;

 loading.value = true;
 try {
  const result = await authApi.login(form.email, form.password);
  // result = { token, user: { id, role } }
  auth.setSession(result.token, result.user);

  if (result.user.role === "student") router.push("/student");
  else if (result.user.role === "teacher") router.push("/teacher");
```

```
    else router.push("/admin");
  } catch (e) {
    error.value = e?.message ?? "Помилка авторизації. Спробуйте ще раз.";
  } finally {
    loading.value = false;
  }
}
</script>

<style scoped>
.page { min-height: 100vh; display: grid; place-items: center; }
.card { width: 420px; border: 1px solid #000; padding: 16px; }
label { display: grid; gap: 6px; margin: 10px 0; }
input { border: 1px solid #000; padding: 8px; }
button { border: 1px solid #000; padding: 10px; width: 100%; }
.error { margin-top: 8px; }
</style>
```

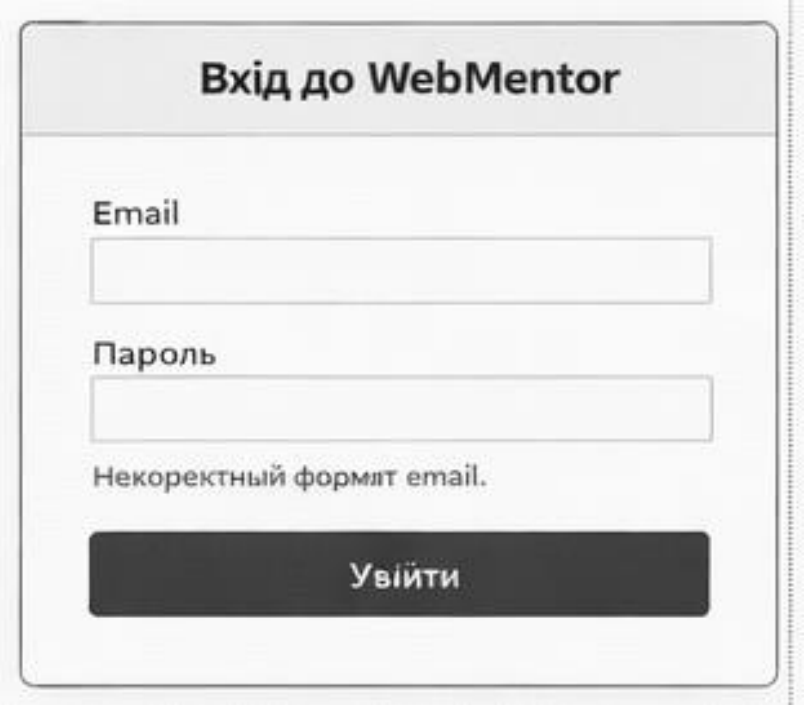

Рисунок 3.2 – Екран входу користувача

Поданий фрагмент програмного коду реалізує екран входу користувача до веб ресурсу WebMentor та забезпечує процес автентифікації в системі (рис. 3.2). Даний інтерфейс відповідає за початкову взаємодію користувача з програмною системою та виконує роль View у межах архітектурного шаблону MVVM. Введені користувачем облікові дані (електронна пошта та пароль) зберігаються у реактивному стані ViewModel, де також реалізується первинна клієнтська валідація. Після підтвердження форми ініціюється подія входу, яка призводить до виклику відповідного API-методу серверної частини системи. Серверна логіка, що відповідає рівню Model, здійснює перевірку облікових даних та

повертає інформацію про користувача. У разі успішної автентифікації клієнтська частина отримує роль користувача та виконує перенаправлення до відповідного головного вікна. Такий підхід забезпечує чіткий розподіл відповідальностей між компонентами View, ViewModel і Model.

Основне вікно студента (Student Dashboard) реалізує (рис. 3.3):

- завантаження списку доступних тестів;
- пошук/фільтрацію;
- перехід до проходження тесту;
- перегляд власних результатів.

Склад програмного модулю у термінах MVVM:

- View: StudentDashboardView.vue;
- ViewModel: стан tests, query, loading, error + метод loadTests();
- Model/API: studentApi.getAvailableTests() + studentApi.getMyResults().

Текст програмного модулю src/views/StudentDashboardView.vue :

```
<template>
 <section class="page">
  <header class="header">
   <h1>Кабінет студента</h1>
   <button @click="logout">Вийти</button>
  </header>

  <div class="toolbar">
   <input v-model.trim="query" placeholder="Пошук тесту..." />
   <button @click="loadTests" :disabled="loading">
    {{ loading ? "Оновлення..." : "Оновити" }}
   </button>
  </div>

  <p v-if="error" class="error">{{ error }}</p>

  <div class="grid">
   <article v-for="t in filteredTests" :key="t.id" class="card">
    <h2>{{ t.title }}</h2>
    <p class="meta">
     Час: {{ t.timeLimitMinutes }} хв • Спроб: {{ t.maxAttempts }}
    </p>
    <button @click="startTest(t.id)">Розпочати</button>
   </article>
  </div>

  <hr />

  <h2>Мої результати</h2>
  <table class="table">
   <thead>
    <tr><th>Тест</th><th>Дата</th><th>Бал</th></tr>
   </thead>
```

```
    <tbody>
      <tr v-for="r in results" :key="r.attemptId">
        <td>{{ r.testTitle }}</td>
        <td>{{ r.finishedAt }}</td>
        <td>{{ r.score }}</td>
      </tr>
    </tbody>
   </table>
  </section>
</template>

<script setup>
import { computed, onMounted, ref } from "vue";
import { useRouter } from "vue-router";
import { studentApi } from "../api/studentApi";
import { useAuthStore } from "../stores/authStore";

const router = useRouter();
const auth = useAuthStore();

const tests = ref([]);
const results = ref([]);
const query = ref("");
const loading = ref(false);
const error = ref("");

const filteredTests = computed(() => {
  const q = query.value.toLowerCase();
  return tests.value.filter(t => t.title.toLowerCase().includes(q));
});

async function loadTests() {
  loading.value = true;
  error.value = "";
  try {
    tests.value = await studentApi.getAvailableTests();
    results.value = await studentApi.getMyResults();
  } catch (e) {
    error.value = e?.message ?? "Не вдалося завантажити дані.";
  } finally {
    loading.value = false;
  }
}

function startTest(testId) {
  router.push(`/student/tests/${testId}`);
}

function logout() {
  auth.clearSession();
  router.push("/login");
}

onMounted(loadTests);
</script>

<style scoped>
.page { padding: 16px; }
.header { display: flex; justify-content: space-between; align-items: center; }
.toolbar { display: flex; gap: 8px; margin: 12px 0; }
input { border: 1px solid #000; padding: 8px; flex: 1; }
button { border: 1px solid #000; padding: 8px 12px; }
```

```
.grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(260px, 1fr)); gap: 12px; }
.card { border: 1px solid #000; padding: 12px; }
.table { width: 100%; border-collapse: collapse; }
.table th, .table td { border: 1px solid #000; padding: 8px; }
.meta { margin: 6px 0 12px; }
</style>
```

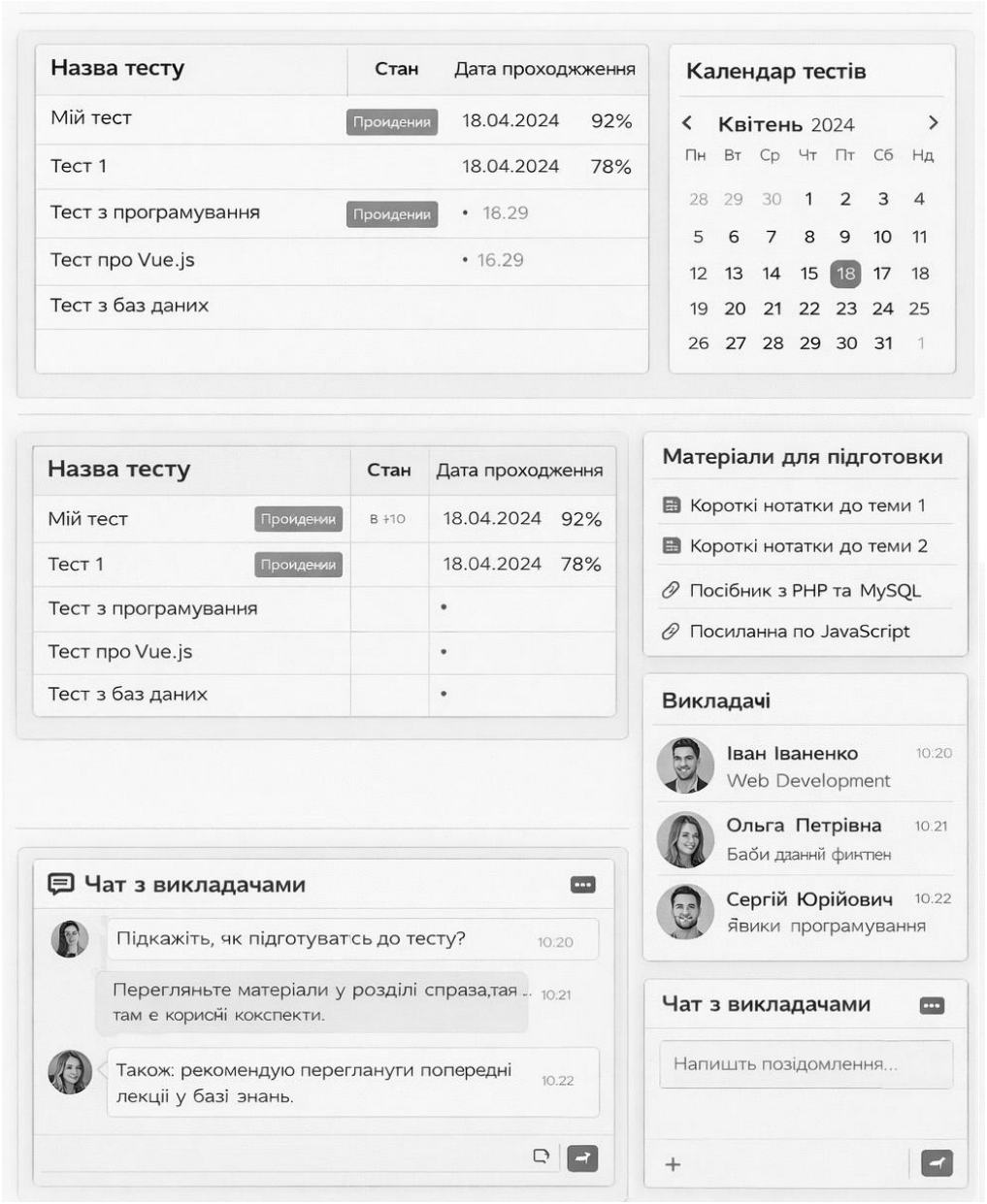

Рисунок 3.3 – Основне вікно студента

Даний інтерфейс представляє основне вікно студента після успішного входу до системи Web Mentor. Компонент виконує функцію View та відповідає за відображення списку доступних тестів, а також результатів проходження попередніх спроб тестування. Дані для відображення формуються у ViewModel, який керує станом інтерфейсу, обробляє події користувача та здійснює фільтрацію інформації. Під час ініціалізації компонента ViewModel викликає

API-методи серверної частини для отримання списку тестів і результатів з бази даних. Серверна частина, що відповідає рівню Model, забезпечує вибірку структурованих даних та їх передачу клієнту. Подія вибору тесту ініціює перехід до модуля проходження тестування. Застосування архітектурного шаблону MVVM дозволяє ізолювати логіку роботи з даними від логіки їх відображення, що підвищує підтримуваність клієнтського коду.

Основне вікно викладача (Teacher Dashboard) забезпечує:

- перегляд власних тестів;
- створення тесту (мінімальний приклад);
- перегляд статистики/результатів по тесту.

Склад програмного модулю у термінах MVVM:

- View: TeacherDashboardView.vue;
- ViewModel: стан myTests, newTest, selectedTestId + методи loadMyTests(), createTest(), loadResults();
- Model/API: teacherApi.*.

Текст програмного модулю src/views/TeacherDashboardView.vue:

```
<template>
 <section class="page">
  <header class="header">
   <h1>Кабінет викладача</h1>
   <button @click="logout">Вийти</button>
  </header>

  <div class="columns">
   <div class="panel">
    <h2>Мої тести</h2>
    <button @click="loadMyTests" :disabled="loading">
     {{ loading ? "Завантаження..." : "Оновити" }}
    </button>

    <ul class="list">
     <li v-for="t in myTests" :key="t.id">
      <button class="link" @click="selectTest(t.id)">
       {{ t.title }}
      </button>
     </li>
    </ul>
   </div>

   <div class="panel">
    <h2>Створити тест</h2>
    <label>
     Назва
     <input v-model.trim="newTest.title" />
    </label>
    <label>
```

```
        Ліміт часу (хв)
        <input v-model.number="newTest.timeLimitMinutes" type="number" min="1" />
      </label>
      <label>
        Кількість спроб
        <input v-model.number="newTest.maxAttempts" type="number" min="1" />
      </label>

      <p v-if="error" class="error">{{ error }}</p>

      <button @click="createTest" :disabled="loadingCreate">
        {{ loadingCreate ? "Створення..." : "Створити" }}
      </button>
    </div>

    <div class="panel">
      <h2>Результати тесту</h2>
      <p v-if="!selectedTestId">Оберіть тест зі списку.</p>

      <table v-else class="table">
        <thead>
          <tr><th>Студент</th><th>Дата</th><th>Бал</th></tr>
        </thead>
        <tbody>
          <tr v-for="r in results" :key="r.attemptId">
            <td>{{ r.studentName }}</td>
            <td>{{ r.finishedAt }}</td>
            <td>{{ r.score }}</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
 </section>
</template>

<script setup>
import { onMounted, ref } from "vue";
import { useRouter } from "vue-router";
import { teacherApi } from "../api/teacherApi";
import { useAuthStore } from "../stores/authStore";

const router = useRouter();
const auth = useAuthStore();

const myTests = ref([]);
const results = ref([]);
const selectedTestId = ref(null);

const newTest = ref({
 title: "",
 timeLimitMinutes: 20,
 maxAttempts: 1
});

const loading = ref(false);
const loadingCreate = ref(false);
const error = ref("");

async function loadMyTests() {
 loading.value = true;
 error.value = "";
```

```
  try {
    myTests.value = await teacherApi.getMyTests();
  } catch (e) {
    error.value = e?.message ?? "Не вдалося завантажити тести.";
  } finally {
    loading.value = false;
  }
}

async function selectTest(testId) {
  selectedTestId.value = testId;
  try {
    results.value = await teacherApi.getTestResults(testId);
  } catch (e) {
    error.value = e?.message ?? "Не вдалося завантажити результати.";
  }
}

function validateNewTest() {
  if (!newTest.value.title) return "Вкажіть назву тесту.";
  if (newTest.value.timeLimitMinutes < 1) return "Ліміт часу має бути > 0.";
  if (newTest.value.maxAttempts < 1) return "Кількість спроб має бути > 0.";
  return "";
}

async function createTest() {
  error.value = validateNewTest();
  if (error.value) return;

  loadingCreate.value = true;
  try {
    await teacherApi.createTest(newTest.value);
    newTest.value.title = "";
    await loadMyTests();
  } catch (e) {
    error.value = e?.message ?? "Не вдалося створити тест.";
  } finally {
    loadingCreate.value = false;
  }
}

function logout() {
  auth.clearSession();
  router.push("/login");
}

onMounted(loadMyTests);
</script>

<style scoped>
.page { padding: 16px; }
.header { display: flex; justify-content: space-between; align-items: center; }
.columns { display: grid; grid-template-columns: 1fr 1fr 1.2fr; gap: 12px; margin-top: 12px; }
.panel { border: 1px solid #000; padding: 12px; }
.list { list-style: none; padding: 0; margin: 10px 0 0; display: grid; gap: 6px; }
.link { border: none; background: none; text-decoration: underline; padding: 0; cursor: pointer; }
input { border: 1px solid #000; padding: 8px; width: 100%; }
label { display: grid; gap: 6px; margin: 10px 0; }
.table { width: 100%; border-collapse: collapse; margin-top: 10px; }
.table th, .table td { border: 1px solid #000; padding: 8px; }
button { border: 1px solid #000; padding: 8px 12px; }
.error { margin-top: 8px; }
```

</style>

Рисунок 3.4 – Основне вікно викладача

На даному рисунку зображено основне вікно викладача вебресурсу WebMentor, призначене для керування тестами та аналізу результатів студентів (рис. 3.4). Інтерфейс виконує роль View та надає засоби для створення нових тестів, перегляду наявних і вибору тесту для аналізу результатів. ViewModel відповідає за зберігання стану форми, перевірку коректності введених даних і оброблення подій, пов’язаних зі створенням тестів. Під час виконання відповідних дій ініціюються API-запити до серверної частини для збереження

або отримання даних. Рівень Model реалізує бізнес-логіку та забезпечує взаємодію з базою даних. Результати тестування студентів відображаються у вигляді таблиці, що формується на основі даних, отриманих із серверної частини. Використання шаблону MVVM забезпечує масштабованість, зручність супроводу та чітку структурованість клієнтської частини веб ресурсу .

API-сервіси (Model / Data layer). Текст модулю src/api/httpClient.js :

```
import { useAuthStore } from "../stores/authStore";

export async function http(url, { method = "GET", body, headers = {} } = {}) {
  const auth = useAuthStore();
  const h = { "Content-Type": "application/json", ...headers };

  if (auth.token) h["Authorization"] = `Bearer ${auth.token}`;

  const res = await fetch(url, {
    method,
    headers: h,
    body: body ? JSON.stringify(body) : undefined
  });

  if (!res.ok) {
    const text = await res.text();
    throw new Error(text || `HTTP ${res.status}`);
  }
  return res.status === 204 ? null : res.json();
}
```

Текст модулю src/api/authApi.js:

```
import { http } from "./httpClient";

export const authApi = {
  login(email, password) {
    return http("/api/auth/login", { method: "POST", body: { email, password } });
  }
};
```

src/api/studentApi.js

```
import { http } from "./httpClient";

export const studentApi = {
  getAvailableTests() {
    return http("/api/student/tests");
  },
  getMyResults() {
    return http("/api/student/results");
  }
};
```

Текст модулю src/api/teacherApi.js :

```
import { http } from "./httpClient";

export const teacherApi = {
  getMyTests() {
    return http("/api/teacher/tests");
  },
  createTest(payload) {
    return http("/api/teacher/tests", { method: "POST", body: payload });
```

```
  },
  getTestResults(testId) {
    return http(`/api/teacher/tests/${testId}/results`);
  }
};
```

У даному підрозділі було розглянуто приклади програмного коду клієнтської частини веб ресурсу WebMentor, які демонструють реалізацію основних користувацьких інтерфейсів системи. Наведені фрагменти коду відображають логіку роботи екрана входу, основного вікна студента та основного вікна викладача, що охоплює ключові сценарії взаємодії користувачів із системою.

Показані приклади підтверджують доцільність використання архітектурного шаблону MVVM, який забезпечує чіткий поділ між рівнями відображення (View), керування станом і логікою інтерфейсу (ViewModel) та оброблення даних (Model). Такий підхід сприяє підвищенню читабельності коду, спрощує його супровід і дозволяє масштабувати клієнтську частину без істотних змін існуючої структури.

Використання API-викликів для взаємодії з серверною частиною забезпечує асинхронний обмін даними та підвищує швидкодію інтерфейсу. Загалом наведені приклади програмного коду демонструють практичну реалізацію теоретичних рішень, закладених на етапі проєктування, та підтверджують ефективність обраного технологічного підходу.

3.4 Реалізація програмного коду для серверної частини веб ресурсу

Серверна частина веб ресурсу WebMentor реалізує бізнес-логіку системи, забезпечує оброблення HTTP-запитів від клієнтської програми та взаємодію з базою даних MySQL. Архітектура серверного компонента побудована за принципом модульного розподілу: окремо виділено маршрутизацію, контролери, сервіси доступу до даних та допоміжні модулі (автентифікація, валідація, формування відповіді). Такий підхід підвищує підтримуваність коду та спрощує розширення системи [12].

Основні функції серверної частини:

1. Автентифікація та авторизація користувачів (адміністратор, викладач, студент).
2. Керування тестами: створення, редагування, публікація, отримання списків тестів.
3. Проходження тестування: створення спроби, збереження відповідей, автоматичне оцінювання, формування результатів.
4. Опитування: створення опитувань, збереження відповідей.
5. Аналітика для викладача: результати за тестом у розрізі груп/студентів.
6. API-взаємодія з клієнтською частиною через JSON (REST-подібний підхід).

   Комунікація з клієнтом відбувається через API-ендпоїнти типу:

   - POST /api/auth/login
   - GET /api/student/tests
   - GET /api/student/results
   - GET /api/teacher/tests
   - POST /api/teacher/tests
   - GET /api/teacher/tests/{id}/results

Приклади коду серверної частини (PHP). Наведені приклади демонструють базову реалізацію API-серверу без прив’язки до конкретного фреймворку (типовий підхід для дипломного проєкту на PHP). За потреби ці приклади легко адаптуються під Laravel/Symfony, але для Web Mentor достатньо модульної структури та PDO.

Структура серверного проєкту, приклад логічної структури:

- public/index.php – точка входу, маршрутизація;
- src/config.php – параметри БД, секрети;
- src/db.php – підключення PDO;
- src/http.php – JSON-відповіді, читання запитів;
- src/auth.php – робота з сесіями/токеном, перевірка ролей;
- src/controllers/AuthController.php;
- src/controllers/StudentController.php;
- src/controllers/TeacherController.php.

Підключення до MySQL через PDO текст модулю src/db.php :

```
<?php
declare(strict_types=1);

function db(): PDO {
   static $pdo = null;
   if ($pdo instanceof PDO) return $pdo;

   $host = DB_HOST;
   $name = DB_NAME;
   $user = DB_USER;
   $pass = DB_PASS;

   $dsn = "mysql:host={$host};dbname={$name};charset=utf8mb4";
   $pdo = new PDO($dsn, $user, $pass, [
      PDO::ATTR_ERRMODE           => PDO::ERRMODE_EXCEPTION,
      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
      PDO::ATTR_EMULATE_PREPARES   => false,
   ]);
   return $pdo;
}
```

Уніфіковані JSON-відповіді та читання тіла запиту, текст модулю src/http.php :

```
<?php
declare(strict_types=1);

function json_input(): array {
   $raw = file_get_contents('php://input');
   if (!$raw) return [];
   $data = json_decode($raw, true);
   return is_array($data) ? $data : [];
}

function json_response(array $data, int $status = 200): void {
   http_response_code($status);
   header('Content-Type: application/json; charset=utf-8');
   echo json_encode($data, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES);
   exit;
}

function error_response(string $message, int $status = 400): void {
   json_response(['error' => $message], $status);
}
```

Автентифікація: вхід користувача (login). У WebMentor пароль зберігається у вигляді хеша (password_hash). Під час входу сервер порівнює введений пароль з хешем та формує токен (або сесію). Для простоти наведено варіант із сесійною авторизацією (підходить для XAMPP/Apache). Текст модулю src/controllers/AuthController.php :

```
<?php
declare(strict_types=1);

require_once __DIR__ . '/../db.php';
require_once __DIR__ . '/../http.php';
```

```
function login_controller(): void {
    $in = json_input();
    $email = trim((string)($in['email'] ?? ''));
    $password = (string)($in['password'] ?? '');

    if ($email === '' || $password === '') {
        error_response('Email і пароль є обов’язковими.', 422);
    }

    $pdo = db();
    $stmt = $pdo->prepare('
        SELECT u.id, u.full_name, u.email, u.password_hash, r.name AS role
        FROM users u
        JOIN roles r ON r.id = u.role_id
        WHERE u.email = :email AND u.status = "active"
        LIMIT 1
    ');
    $stmt->execute(['email' => $email]);
    $user = $stmt->fetch();

    if (!$user || !password_verify($password, $user['password_hash'])) {
        error_response('Невірні облікові дані.', 401);
    }

    // Сесія
    session_start();
    $_SESSION['uid']  = (int)$user['id'];
    $_SESSION['role'] = (string)$user['role'];

    json_response([
        'user' => [
            'id' => (int)$user['id'],
            'fullName' => (string)$user['full_name'],
            'email' => (string)$user['email'],
            'role' => (string)$user['role'],
        ]
    ]);
}
```

Перевірка доступу за ролями, модуль src/auth.php :

```
<?php
declare(strict_types=1);

require_once __DIR__ . '/http.php';

function require_auth(): array {
    session_start();
    $uid = $_SESSION['uid'] ?? null;
    $role = $_SESSION['role'] ?? null;

    if (!$uid || !$role) {
        error_response('Потрібна авторизація.', 401);
    }
    return ['uid' => (int)$uid, 'role' => (string)$role];
}

function require_role(string ...$allowed): array {
    $auth = require_auth();
    if (!in_array($auth['role'], $allowed, true)) {
        error_response('Недостатньо прав доступу.', 403);
```

```
    }
    return $auth;
}
```

API для студента: список доступних тестів та результати проходження. Сервер повертає лише опубліковані тести (параметр is_published = 1), а також базові налаштування тесту. Модуль src/controllers/StudentController.php :

```
<?php
declare(strict_types=1);

require_once __DIR__ . '/../db.php';
require_once __DIR__ . '/../http.php';
require_once __DIR__ . '/../auth.php';

function student_tests_controller(): void {
    require_role('student');

    $pdo = db();
    $stmt = $pdo->query('
        SELECT id, title, time_limit_minutes, max_attempts
        FROM tests
        WHERE is_published = 1
        ORDER BY created_at DESC
    ');
    $tests = $stmt->fetchAll();

    json_response(['tests' => $tests]);
}

function student_results_controller(): void {
    $auth = require_role('student');

    $pdo = db();
    $stmt = $pdo->prepare('
        SELECT a.id AS attempt_id, t.title AS test_title, a.finished_at, a.score
        FROM attempts a
        JOIN tests t ON t.id = a.test_id
        WHERE a.student_id = :sid AND a.status = "finished"
        ORDER BY a.finished_at DESC
        LIMIT 50
    ');
    $stmt->execute(['sid' => $auth['uid']]);

    json_response(['results' => $stmt->fetchAll()]);
}
```

API для викладача: створення тесту, модуль src/controllers/TeacherController.php:

```
<?php
declare(strict_types=1);

require_once __DIR__ . '/../db.php';
require_once __DIR__ . '/../http.php';
require_once __DIR__ . '/../auth.php';
```

```
function teacher_create_test_controller(): void {
   $auth = require_role('teacher');

   $in = json_input();
   $title = trim((string)($in['title'] ?? ''));
   $time = (int)($in['timeLimitMinutes'] ?? 0);
   $attempts = (int)($in['maxAttempts'] ?? 0);

   if ($title === '' || $time <= 0 || $attempts <= 0) {
      error_response('Некоректні параметри тесту.', 422);
   }

   $pdo = db();
   $stmt = $pdo->prepare('
      INSERT INTO tests (teacher_id, title, description, time_limit_minutes, max_attempts, is_published, created_at)
      VALUES (:tid, :title, "", :time, :attempts, 0, NOW())
   ');
   $stmt->execute([
      'tid' => $auth['uid'],
      'title' => $title,
      'time' => $time,
      'attempts' => $attempts
   ]);

   json_response(['testId' => (int)$pdo->lastInsertId()], 201);
}
```

Маршрутизація (точка входу), модуль public/index.php:

```
<?php
declare(strict_types=1);

require_once __DIR__ . '/../src/config.php';
require_once __DIR__ . '/../src/http.php';

require_once __DIR__ . '/../src/controllers/AuthController.php';
require_once __DIR__ . '/../src/controllers/StudentController.php';
require_once __DIR__ . '/../src/controllers/TeacherController.php';

$method = $_SERVER['REQUEST_METHOD'] ?? 'GET';
$path = parse_url($_SERVER['REQUEST_URI'] ?? '/', PHP_URL_PATH);

if ($method === 'POST' && $path === '/api/auth/login') {
   login_controller();
}

if ($method === 'GET' && $path === '/api/student/tests') {
   student_tests_controller();
}

if ($method === 'GET' && $path === '/api/student/results') {
   student_results_controller();
}

if ($method === 'POST' && $path === '/api/teacher/tests') {
   teacher_create_test_controller();
}

error_response('Endpoint not found', 404);
```

У серверній частині доцільно зазначити такі заходи:

- використання PDO prepared statements для захисту від SQL-ін’єкцій;
- зберігання паролів як password_hash() + перевірка password_verify();
- контроль доступу через рольову модель (RBAC) на рівні API;
- серверна валідація вхідних даних (навіть якщо є клієнтська);
- обмеження помилок/логування без розкриття конфіденційної інформації користувачу.

Серверна частина веб ресурсу WebMentor реалізує основну бізнес-логіку системи та забезпечує взаємодію клієнтської програми з базою даних. Побудова серверного компонента на основі мови програмування PHP і системи керування базами даних MySQL дозволила реалізувати надійний механізм оброблення запитів, зберігання даних і контролю доступу користувачів.

Використання модульної структури та REST-подібного підходу до організації API сприяє зручності розширення і супроводу програмного забезпечення. Реалізовані механізми автентифікації, рольового доступу та валідації вхідних даних підвищують рівень безпеки системи. Загалом серверна частина забезпечує стабільне функціонування веб ресурсу та є надійною основою для реалізації процесів тестування й опитувань у навчальному середовищі.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто поставленої мети, а саме, розроблено веб ресурс Web Mentor, призначений для організації та проведення тестування й опитувань у навчальному процесі. Розроблений програмний продукт орієнтований на використання в умовах як очного, так і дистанційного навчання та забезпечує автоматизований контроль рівня знань здобувачів освіти.

У першому розділі роботи проаналізовано актуальні проблеми організації контролю знань у сучасному освітньому середовищі, а також розглянуто наявні програмні рішення для тестування та опитувань. Проведений порівняльний аналіз популярних систем перевірки знань дозволив визначити їх переваги та недоліки і сформувати обґрунтовані вимоги до функціональних можливостей розроблюваного веб ресурсу.

У другому розділі було обґрунтовано вибір технологій і засобів розроблення, зокрема використання мов програмування JavaScript і PHP, фреймворку Vue.js, веб сервера Apache та системи керування базами даних MySQL. Також було спроєктовано архітектуру веб ресурсу, визначено взаємодію між клієнтською та серверною частинами, а також розроблено структуру бази даних з урахуванням принципів нормалізації та цілісності даних.

У третьому розділі описано програмну реалізацію веб ресурсу Web Mentor. Розглянуто реалізацію клієнтської частини з використанням архітектурного шаблону MVVM, що забезпечує чіткий поділ логіки відображення, керування станом і роботи з даними. Наведено приклади програмного коду серверної частини, яка реалізує бізнес-логіку системи, механізми автентифікації, рольового доступу та взаємодії з базою даних. Особливу увагу приділено питанням безпеки та коректності оброблення вхідних даних. Також, продемонстровано функціонування розробленого веб ресурсу з точки зору користувача. Описано сценарії роботи для студентів і викладачів, наведено приклади інтерфейсів та проєкт алгоритму роботи системи. Реалізовані інструменти дозволяють викладачам створювати та адмініструвати тести,

аналізувати результати, а студентам проходити тестування, переглядати власні результати та отримувати навчальні матеріали.

Результати виконаної роботи підтверджують, що розроблений веб ресурс Web Mentor є функціонально завершеним, зручним у використанні та придатним для впровадження в освітній процес. Запропоноване рішення сприяє підвищенню об’єктивності, прозорості та ефективності контролю знань, а також може бути розширене шляхом додавання нових модулів і функціональних можливостей у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке Moodle. URL: https://moodle.org/login/index.php?loginredirect=1 (дата звернення 15. 02. 2023).
2. Moodle Docs. Moodle Learning Management System. URL: https://docs.moodle.org/ (дата звернення: 10. 02. 2023).
3. Биков В. Ю. Інновації в організації досліджень та розробок у галузі інформаційно-комунікаційних технологій в освіті у світлі викликів XXI сторіччя. URL: http://appsychology.org.ua/data/jrn/v8/i10/7.pdf (дата звернення: 10. 02. 2023).
4. Морзе Н. В., Кузьмінська О. Г. Педагогічні аспекти використання інформаційно-комунікаційних технологій. К. : Педагогічна думка, 2020. 198 с.
5. Flanagan D. JavaScript: The Definitive Guide. – 7th ed. – Sebastopol : O'Reilly Media, 2020. 706 p. URL: https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/ (date of access: 09. 03. 2023).
6. Banks A., Porcello E. Learning React / Modern JavaScript Frameworks. – Sebastopol : O'Reilly Media, 2020. 350 p. URL: https://surl.li/cxxmlw (date of access: 14. 03. 2023).
7. PHP Documentation. PHP Manual. URL: https://www.php.net/manual/ (дата звернення: 15. 03. 2023).
8. MySQL Documentation. MySQL Reference Manual. URL: https://dev.mysql.com/doc/ (дата звернення: 15. 03. 2023).
9. Запуск та керування Apache URL: https://kovelpost.com/blogs/9353 (дата звернення 15.02.2023).
10. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). URL: https://www.iso.org/standard/35733.html (date of access: 14. 03. 2023).
11. Richardson C. Microservices Patterns. – New York : Manning Publications, 2019. 520 p. URL: https://surl.lt/nlndry (date of access: 14. 03. 2023).

12. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. – University of California, Irvine, 2000. URL: https://surli.cc/nskdae (date of access: 14. 03. 2023).