

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інформаційна система з підтримки дистанційної
форми навчання для
загальноосвітньої школи (робота з користувачами,
створення тестування)»**

Виконала студентка групи ІІЗ-19-1
АНДРУЩЕНКО Людмила Валеріївна

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **АНДРУЩЕНКО Людмила Валеріївна**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи (робота з користувачами, створення тестування)»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2023 р., №70с**

Вихідні дані для роботи: Система, яка надає підтримку та забезпечує ефективне функціонування дистанційного навчання в загальноосвітніх школах. А також застосування сучасних веб-технологій при розробці системи.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.,	17.02.2023р.	17.02.2023р.
2	Вакалюк Т.А.,	12.04.2023р.	12.04.2023р.
3	Вакалюк Т.А.,	02.05.2023р.	02.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналізування вимог до програмного продукту	14 лютого 2023 – 28 лютого 2023	Виконано
2	Пошук, огляд та аналіз аналогічних веб-сайтів	1 березня 2023 – 15 березня 2023	Виконано
3	Вивчення матеріалу по конкретній тематиці	16 березня 2023 – 25 березня 2023	Виконано
4	Планування структури програми	26 березня 2023 – 30 квітня 2023	Виконано
5	Проектування та реалізація архітектури веб-системи кінотеатру	1 квітня 2023 – 4 квітня 2023	Виконано
6	Написання програмного коду	10 квітня 2023 – 14 червня 2023	Виконано
7	Тестування роботи програми	14 червня 2023 – 15 червня 2023	Виконано
8	Аналіз одержаних результатів	15 червня 2023 – 16 червня 2023	Виконано
9	Оформлення пояснювальної записки	9 червня 2023 – 16 червня 2023	Виконано
10	Попередній захист	26 квітня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу інформаційної системи з підтримки дистанційної форми навчання для загальноосвітньої школи та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 63 сторінках, має 25 сторінок додатків. Список використаних джерел містить 25 найменувань і займає 2 сторінки. В роботі наведено 36 рисунків та 8 таблиць. Загальний обсяг роботи – 91 сторінка.

Метою роботи є розробка інформаційної системи з підтримки дистанційної форми навчання для загальноосвітньої школи, створення ефективного та зручного інструменту, який забезпечує взаємодію між користувачами системи (учнями, вчителями та адміністраторами) та сприяє покращенню процесу навчання та оцінювання знань.

У роботі були сформульовані основні завдання, що передбачали розробку інформаційної системи, проведено аналіз аналогів програмного продукту, детально обгрунтовано архітектуру проекту і наведено алгоритми його роботи. Крім того, були представлені приклади роботи програми та виконано тестування для перевірки її функціональності та надійності.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ФРЕЙМВОРК, БІБЛІОТЕКА, СИСТЕМА.

					ІПЗ.КР.Б-121-23-ІПЗ						
Зм.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Л.В.Андрущенко			Інформаційна система з підтримки дистанційної форми навчання для загальноосвітньої школи (робота з користувачами, створення тестування)	Літ.			Арк.	Аркуші	
Керівник		Т.А.Вакалюк							2		
						Житомирська політехніка, група ІПЗ-19-1					
н. контр.		О.П. Тичина									
Зав. каф.		А.В.Морозов				Пояснювальна записка					

ABSTRACT

The bachelor's graduation consists of a software complex "Information System for Supporting Distance Learning in a Secondary School" and an explanatory note. The explanatory note for the thesis includes a textual part presented on 63 pages and comprises 25 pages of appendices. The list of used sources contains 25 references and occupies 2 pages. The thesis includes 36 figures and 8 tables. The total volume of the thesis is 91 pages.

The purpose of the graduation work is to develop an information system that supports distance learning in a secondary school and creates an efficient and convenient tool. This system facilitates interaction among system users (students, teachers, and administrators) and improves the learning process and knowledge assessment.

The main tasks for the thesis included formulating the development objectives of the information system, conducting an analysis of similar software products, providing a detailed justification for the project's architecture, and presenting the algorithms of its operation. Additionally, program examples were demonstrated, and testing was conducted to verify its functionality and reliability.

KEYWORDS: WEB APPLICATION, FRAMEWORK, LIBRARY, SYSTEM.

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		3

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ	8
1.1 Постановка задачі.....	8
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури інформаційної системи	16
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	18
Висновки до першого розділу.	21
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ.....	22
2.1 Визначення варіантів використання і структури додатку	22
2.3 Проектування та реалізація алгоритмів роботи додатку	35
2.4 Реалізація функціоналу системи з підтримки дистанційної форми навчання для загальноосвітньої школи	38
Висновки до другого розділу	43
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ	45
3.1 Порядок встановлення та налаштування параметрів системи	45
3.2 Структура інтерфейсу системи з підтримки дистанційної форми навчання для загальноосвітньої школи	47
3.3 Тестування роботи додатку	56
Висновки до третього розділу	59
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	62
ДОДАТКИ.....	64

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		4

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

API – Прикладний програмний інтерфейс

SPA – Single page application, архітектура веб-додатків, де весь додаток міститься на одній веб-сторінці

MVC – Model-View-Controller, це архітектурний шаблон, що використовується для розробки програмного забезпечення

ПЗ – програмне забезпечення

UI – інтерфейс користувача

		Л.В.Андрущенко			ПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		5

ВСТУП

Актуальність теми. Швидкий розвиток інформатизації суспільства та все більше використання Інтернету призвели до появи дистанційного навчання як нової форми освітнього процесу. Цей метод навчання, що поєднує традиційні методи з новими інформаційними та телекомунікаційними технологіями, виявився перспективним, гуманістичним та індивідуалізованим.

Дистанційне навчання забезпечує більш зручний та швидкий доступ до знань завдяки використанню навчальних ресурсів через Інтернет та відсутності необхідності у фізичних приміщеннях. Цей підхід також сприяє легкому оновленню та коригуванню електронних навчальних матеріалів та забезпечує можливість одночасного доступу для багатьох користувачів.

Таким чином, актуальність обраної теми полягає в потребі в розробці інформаційної системи, яка підтримує дистанційну форму навчання, враховуючи швидкий розвиток технологій та зміни в освітній практиці.

Метою випускної роботи є розробка та проектування інформаційної системи з підтримки дистанційної форми навчання для загальноосвітньої школи, створення ефективного та зручного інструменту, який забезпечує взаємодію між користувачами системи та сприяє покращенню процесу навчання та оцінювання знань.

Встановлена мета обумовлює наступні завдання:

- проведення аналізу функціональних потреб для інформаційної системи, що підтримує дистанційну форму навчання;
- збір і аналіз інформації про аналогічні продукти на ринку та їх порівняння;
- обґрунтування і вибір необхідних засобів для реалізації додатку, враховуючи вимоги та потреби користувачів;
- вибір архітектури веб-застосунку, що забезпечує оптимальну роботу та ефективне використання ресурсів;
- розробка самої інформаційної системи та реалізація всіх визначених функціональних вимог, забезпечуючи їхню повноту та надійність.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	6
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Об’єктом дослідження є система, яка надає підтримку та забезпечує ефективне функціонування дистанційного навчання в загальноосвітніх школах.

Предметом дослідження є застосування сучасних веб-технологій при розробці системи для дистанційної форми навчання в загальноосвітніх школах.

В роботі було використано методи аналізу функціональних потреб користувача, методи програмування веб-застосунків за допомогою фреймворку Laravel та бібліотеки React та методи дизайну.

Даний продукт може бути використаний в загальноосвітніх школах для забезпечення дистанційного навчання. Він дозволяє створювати предмети, надавати учням доступ до навчальних матеріалів, проводити тестування та оцінювання. Також система допомагає вчителям відстежувати прогрес учнів, спрощує організацію навчального процесу. Застосування цього продукту дозволяє покращити навчальний процес, забезпечуючи зручність, гнучкість та ефективність дистанційного навчання.

Пояснювальна записка до випускної роботи містить 63 сторінки, 36 ілюстрацій та 8 таблиць, список літературних джерел містить 25 найменувань.

Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 25 найменувань (2 стор.), 1 додатку (25 стор.), 8 таблиць та 36 ілюстрацій по тексту. Повний обсяг роботи становить 91 сторінку, з них 63 сторінки основного тексту.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	7
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ

1.1 Постановка задачі

Метою розробки є створення веб-застосунку, який буде призначений для навчальних закладів з метою спрощення створення занять, додавання завдань до них та перевірки їх безпаперовим шляхом, а також створення тестувань. За допомогою системи є можливість перегляду журналу для кожного класу здобувачів освіти. Сервіс дає можливість створення облікових записів для учнів, вчителів та адміністраторів, які використовують частковий або повний функціонал системи відповідно. Основна мета системи — прискорити процес навчання.

Тож основними етапами розробки даної системи є:

1. Аналіз вимог.
 - 1.1.Збір вимог користувачів до системи.
 - 1.2.Аналіз функціональних, технічних та інших вимог.
2. Проектування архітектури.
 - 2.1.Визначення загальної архітектури системи.
 - 2.2.Визначення компонентів системи та їх взаємодії.
 - 2.3.Вибір технологій та інструментів.
3. Розробка дизайну користувацького інтерфейсу.
 - 3.1.Визначення загальної концепції дизайну системи.
 - 3.2.Розробка дизайну сторінок та компонентів для користувачів, зокрема:
 - 3.2.1. дизайн панелі головного меню;
 - 3.2.2. дизайн сторінки профілю користувача;
 - 3.2.3. дизайн сторінки списку предметів, тестів, класів та користувачів;

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		8

3.2.4. дизайн сторінки створення та редагування тестів;

3.2.5. дизайн сторінки відображення результатів тестування.

4. Розробка бази даних.

4.1.Визначення структури бази даних з урахуванням потреб системи.

4.2.Розробка таблиць та взаємозв'язків між ними.

5. Розробка функціоналу користувачів:

5.1.Розробка функціоналу для створення/видалення/редагування користувачів.

5.2.Реалізація механізмів перевірки та підтвердження даних користувачів.

5.3.Розділення ролей користувачів.

6. Розробка системи реєстрації та автентифікації користувачів:

6.1.розробка сторінок та компонентів для реєстрації та входу користувачів;

6.2.розробка функціоналу перевірки даних користувачів та керування доступом;

7. Розробка функціоналу створення тестів.

7.1.Розробка сторінок та компонентів для створення/видалення/редагування тестів.

7.2.Розробка функціоналу додавання та редагування питань тесту.

7.3.Розробка функціоналу для проходження тестувань учнями.

8. Розробка функціоналу класів:

8.1.розробка функціоналу створення/видалення/редагування класів;

8.2.реалізація механізмів призначення вчителів та учнів до класів;

9. Тестування та налагодження системи:

9.1.функціональне тестування розробленого функціоналу;

9.2.тестування графічного інтерфейсу та користувацького досвіду.

У майбутньому додатку повинні бути такі ролі користувачів: учні, вчителі та адміністратори, нижче наведені їхні можливості.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	9
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Адміністратор

1. Можливість створення користувачів.
2. Можливість створення шкільних предметів.
3. Можливість створення класу учнів.
4. Можливість додання до класу самих учнів та класного керівника.
5. Можливість переглядати журнали успішності студентів.

Вчитель

1. Має змогу додавати завдання до відповідного предмету.
2. Має змогу створювати тести для перевірки знань учнів.
3. Має змогу перевіряти виконані завдання та виставляти оцінки.
4. Має можливість переглядати журнали успішності студентів за предметами, котрі викладає.

Учень

1. Можливість перегляду та виконання завдань.
2. Можливість проходження тестів задля перевірки знань.
3. Можливість перегляду власної успішності.

1.2 Аналіз аналогів програмного продукту

В сьогоденні існує багато платформ для онлайн навчання, тож проаналізуємо найпопулярніші з них:

- HUMAN ШКОЛА [1];
- MOODLE [2];
- GOOGLE CLASSROOM [3].

HUMAN ШКОЛА

HUMAN Школа - українська освітня платформа для закладів освіти, що об'єднує учасників освітнього процесу в єдиному захищеному середовищі.

Основні функції HUMAN ШКОЛА включають:

- управління класами: вчителі можуть створювати класи та додавати учнів до них. Вони можуть організовувати розклад занять, призначати домашні

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	10
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

завдання та стежити за академічним прогресом учнів;

- електронний журнал: Платформа HUMAN ШКОЛА дозволяє вчителям вести електронний журнал, в якому можна виставляти оцінки, вносити коментарі та відстежувати успішність учнів. Батьки також можуть отримувати доступ до журналу, щоб бути в курсі академічного прогресу своїх дітей;
- онлайн-класи: HUMAN ШКОЛА надає можливість проводити онлайн-уроки та віртуальні заняття. Вчителі можуть використовувати інтерактивні інструменти, ділитися екраном, завантажувати матеріали для навчання та взаємодіяти з учнями у режимі реального часу;
- оцінювання та тестування: HUMAN ШКОЛА дозволяє вчителям створювати тести та оцінювати знання учнів. Вони можуть створювати різні типи питань, встановлювати терміни тестування та аналізувати результати;
- комунікація: платформа надає засоби для спілкування між вчителями, учнями та батьками. Вони можуть обмінюватися повідомленнями, завданнями, матеріалами та інформацією про навчальний процес;
- моніторинг академічного прогресу: HUMAN ШКОЛА дозволяє адміністраторам та вчителям відстежувати академічний прогрес учнів, аналізувати результати тестів та оцінок, та надавати індивідуальну підтримку для покращення успішності учнів.

і простори

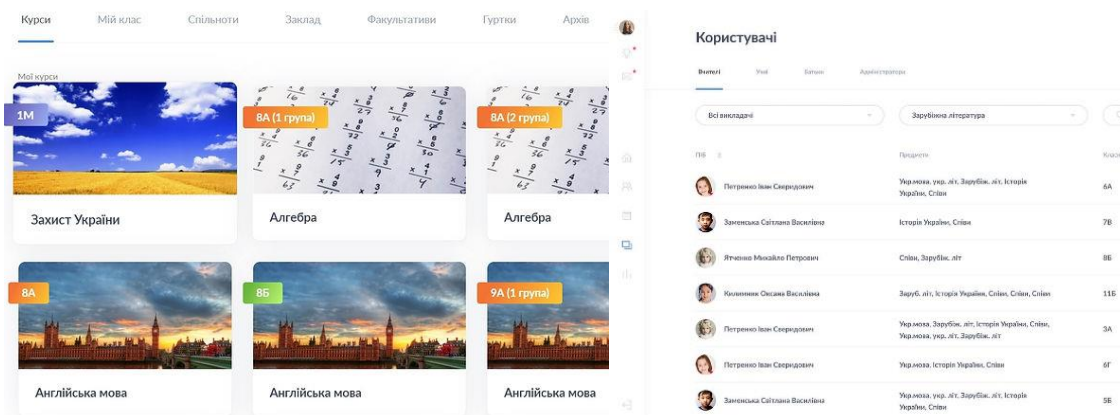


Рисунок 1.1 – Зовнішній вигляд платформи Human Школа.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	11
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

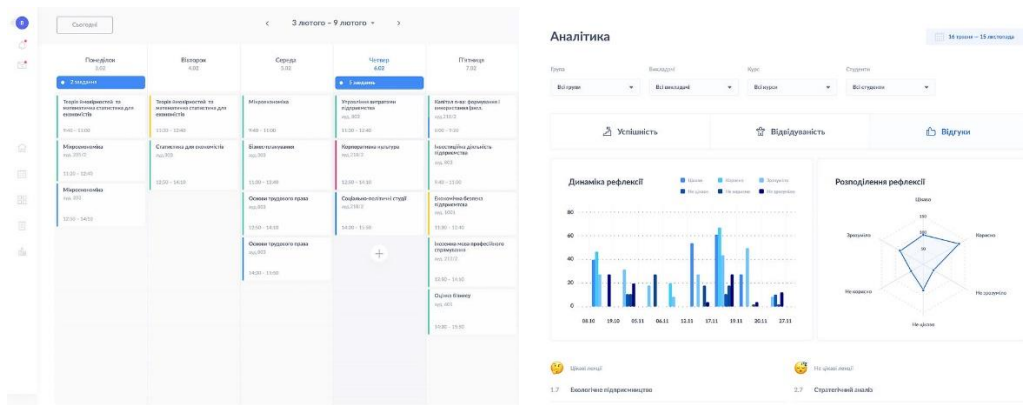


Рисунок 1.2 – Зовнішній вигляд платформи Human Школа.

Переваги HUMAN ШКОЛА:

- широкий функціонал;
- зручний та зрозумілий інтерфейс користувача, що спрощує навігацію та використання системи;
- високий рівень безпеки та конфіденційності даних.

Недоліки HUMAN ШКОЛА:

- можливість перевантаження системи при великій кількості користувачів або обсязі даних.

MOODLE

MOODLE – навчальна платформа, призначена для об'єднання педагогів, адміністраторів і учнів в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища.

Основні функції Moodle включають:

- управління курсами: moodle дозволяє викладачам створювати та керувати курсами. Вони можуть створювати структуру курсу, додавати матеріали, завдання, відео, посилання та інші ресурси;
- взаємодія та спілкування: moodle надає засоби для спілкування та взаємодії між викладачами та учнями. Це включає форуми, чати, блоги, відкриті обговорення, електронну пошту та повідомлення в системі. Викладачі можуть стежити за прогресом учнів, спілкуватися з ними та надавати необхідну підтримку;

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	12
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

- завдання та тестування: moodle має можливості для створення завдань та тестів для оцінювання знань учнів. Викладачі можуть створювати різновиди завдань, такі як написання есе, завдання з вибором, завдання з налагодження коду тощо. Також є можливість проведення онлайн-тестів з автоматичною перевіркою результатів;
- оцінювання та звіти: moodle дозволяє викладачам виставляти оцінки за завдання, тести та інші види робіт. Викладачі можуть стежити за успішністю учнів, генерувати звіти та аналізувати прогрес кожного учня окремо або в цілому для всього курсу;
- доступ до матеріалів: moodle надає можливість завантажувати та спільно використовувати різноманітні матеріали, такі як презентації, документи, відео, аудіо-файли, веб-посилання та інші ресурси. Це дозволяє викладачам надати учням необхідний навчальний матеріал для вивчення;
- адаптивний дизайн: moodle підтримує адаптивний дизайн, що дозволяє користувачам зручно працювати з платформою на різних пристроях, включаючи комп'ютери, планшети та смартфони.
- розширення та інтеграції: moodle може бути розширений за допомогою додаткових модулів та плагінів, що дозволяє додати нові функціональні можливості.

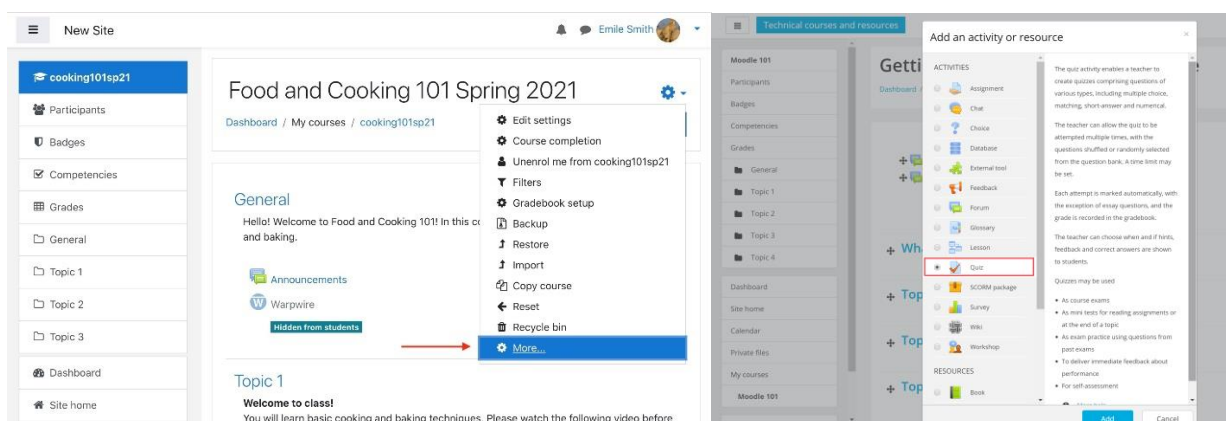


Рисунок 1.3 – Зовнішній вигляд платформи MOODLE.

Переваги MOODLE:

- зручна навігація;
- онлайн спілкування;

		Л.В.Андрущенко			ПЗ.КР.Б-121-23-ПЗ	13
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

- зручний доступ до матеріалів;
- оцінювання та звітність.

Недоліки MOODLE:

- велика кількість функцій (іноді користувачі можуть виявити, що це може бути певною мірою заплутаним);
- потребує навчання;
- потребує підтримки адміністратора.

GOOGLE CLASSROOM

Google Classroom — безкоштовний веб-сервіс, створений Google для навчальних закладів з метою спрощення створення, поширення і класифікації завдань безпаперовим шляхом. Основна мета сервісу — прискорити процес поширення файлів між педагогами та здобувачами освіти.

Основні функції Google Classroom включають:

- створення класів: вчителі можуть створювати окремі класи для кожного предмета або групи учнів. Кожен клас має власний унікальний код, за допомогою якого учні можуть приєднатися до класу;
- додавання учнів та вчителів: вчителі можуть запрошувати учнів та інших вчителів до своїх класів, використовуючи електронну пошту або код класу. Це спрощує процес включення учасників до навчання;
- публікація завдань: вчителі можуть створювати завдання, вказувати терміни виконання та прикріплювати необхідні матеріали, такі як тексти, фотографії або посилання. Учні можуть легко отримати доступ до завдань та виконувати їх;
- завершення та відправка завдань: учні можуть виконувати завдання, редагувати тексти, додавати відповіді та відправляти їх для перевірки вчителем. Вчителі можуть оцінювати завдання та надавати зворотний зв'язок;
- обмін матеріалами: вчителі можуть надсилати матеріали, такі як презентації, документи або посилання, безпосередньо до класу. Учні

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	14
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

можуть отримувати ці матеріали та зберігати їх для подальшого використання;

- обговорення та спілкування: учні та вчителі можуть взаємодіяти через обговорення в рамках класу. Вони можуть обмінюватися думками, задавати питання та отримувати відповіді, що сприяє активній комунікації в класі;
- графік та повідомлення: вчителі можуть створювати розклад занять та нагадування для учнів. Крім того, вони можуть надсилати повідомлення або сповіщення про оновлення до всього класу або окремим учням.

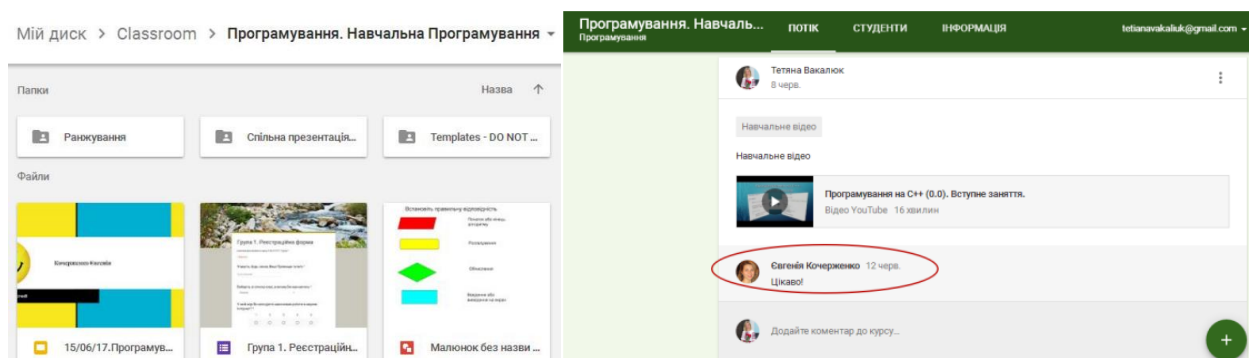


Рисунок 1.4 – Зовнішній вигляд платформи Google Classroom.

Переваги Google Classroom:

- простий інтуїтивно зрозумілий інтерфейс;
- легке створення та організація класів і завдань;
- швидка і проста спільна робота над документами;
- інтеграція з іншими інструментами Google, такими як Google Drive та Google Календар.

Недоліки Google Classroom:

- обмежені можливості в порівнянні з іншими системами керування навчанням;
- відсутність деяких розширених функцій, які можуть бути потрібні для деяких освітніх сценаріїв;
- обмежена підтримка для багатомовності та локалізації.

Отож, у процесі аналізу аналогічних додатків і враховуючи найкращі практики в галузі, власна система пропонуватиме широкий спектр

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	15
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

функціональних можливостей, що дозволяють ефективно організувати навчальний процес, матиме вбудовану систему оцінювання та відстеження прогресу студентів. Викладачі можуть створювати тести, завдання та оцінювати роботи студентів. Також платформа повинна бути простою у використанні, з мінімалістичним дизайном та зрозумілим інтерфейсом.

1.3 Вибір архітектури інформаційної системи

Вибір архітектури додатку є критично важливим етапом у процесі розробки, оскільки він має великий вплив на ефективність, розширюваність та підтримку проекту у майбутньому. Вірно обрана архітектура забезпечує стабільну та зручну основу для розробки, тестування, розширення та підтримки додатку протягом його життєвого циклу.

Наш програмний комплекс повинен бути побудований на клієнт-серверній архітектурі з використанням веб-технологій, що не вимагають додаткового ліцензування.

На серверній частині додатку буде доцільно використати архітектуру, яка оснований на MVC [4]. Вищезазначена аббревіатура розшифровується як Model - View - Controller.

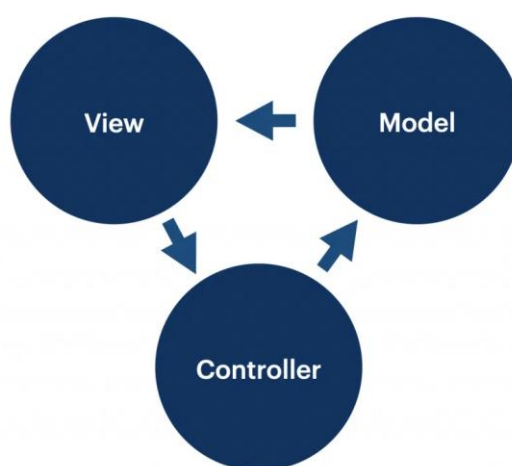


Рисунок 1.5 – Відображення архітектурного шаблону MVC.

Вона дозволяє чітко розділити різні компоненти системи, забезпечуючи гнучкість та легкість у розробці. Модель(Model) представляє бізнес-логіку та

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	16
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

взаємодію з базою даних, представлення(View) відповідає за відображення даних та взаємодію з користувачем, а контролер(Controller) керує потоком даних та взаємодіє з моделлю та представленням. Це дозволяє розробникам працювати над різними компонентами незалежно один від одного, сприяє повторному використанню коду та забезпечує зрозумілість та структурованість проекту.

Клієнтська частина повинна бути побудована за відомою архітектурою SPA [5]. Single Page Application, або «додаток однієї сторінки» – це тип веб-додатків, в яких завантаження необхідного коду відбувається на одну сторінку. SPA має кілька ключових переваг, які сприяють покращенню користувацького досвіду (UX) та забезпечують ефективну роботу веб-додатків. Миттєве завантаження даних: SPA завантажує всі необхідні ресурси під час першого запуску і потім працює без перезавантаження сторінки. Це дозволяє швидко відображати зміни на сторінці, оновлювати дані без необхідності перезавантаження та забезпечує плавну і неперервну роботу додатка. Покращений UX: SPA надає більш інтерактивний та зручний користувацький досвід. Він дозволяє впроваджувати анімацію, переходи між сторінками без затримок, динамічне оновлення контенту та реагувати на взаємодію користувача швидко та плавно. Швидка робота на мобільних пристроях: SPA оптимізований для роботи на мобільних пристроях, де швидкість і ефективність важливі. Він забезпечує меншу кількість запитів до сервера, мінімізує трафік даних та робить додаток більш реагуючим на дотики та жести. Кросплатформність: SPA дозволяє розробляти веб-додатки, які працюють на різних платформах, включаючи комп'ютери, планшети та смартфони. Це зменшує необхідність розробляти окремі версії для кожної платформи та спрощує підтримку та оновлення додатку. Швидкість та масштабованість: SPA дозволяє зменшити навантаження на сервер та оптимізувати завантаження сторінок. Він дозволяє використовувати кешування, асинхронні запити та інші техніки, що забезпечують швидку роботу додатка, особливо при великій кількості користувачів.

MVC на бекенді та SPA на фронтенді – таке поєднання надасть нам можливості для ефективної розробки, зручного користування та майбутнього

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	17
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

розширення нашої власної системи у відповідності з подальшими цілями та вимогами.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Після проведення аналізу аналогів веб-застосунків, з'ясування потрібних вимог до нашого додатку та вибору архітектури, наступним кроком є обґрунтування вибору інструментальних засобів та встановлення вимог до апаратного забезпечення.

Для розробки серверної частини нашого веб-застосунку ми обрали веб-сервер Apache2, який є одним з найпопулярніших та надійних веб-серверів. Він підтримує HTTP та HTTPS [6] протоколи і забезпечує надійну доставку веб-сторінок клієнтам.

В якості системи керування базами даних ми вибрали MySQL [7], оскільки вона є потужною, широко підтримується та має багатий функціонал. Використовується PHPMyAdmin [8] для керування базами даних. Вибір PHPMyAdmin є обґрунтованим з таких причин:

- легкість використання: PHPMyAdmin надає зручний інтерфейс користувача, що дозволяє зручно керувати базами даних. Ви можете створювати, редагувати та видаляти таблиці, запити і інші елементи бази даних без необхідності писати складні SQL-запити;
- функціональність: PHPMyAdmin надає широкий набір функцій, таких як імпорт та експорт даних, зміна структури таблиць, виконання складних запитів та багато іншого. Це дозволяє зручно працювати з базами даних і виконувати потрібні операції безпосередньо з веб-інтерфейсу;
- підтримка MySQL: PHPMyAdmin спеціально розроблений для роботи з MySQL, однією з найпопулярніших систем керування базами даних. Це означає, що ви зможете ефективно керувати вашими MySQL-базами даних і виконувати всі необхідні дії, пов'язані з ними.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	18
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Для розробки серверної частини ми використовуємо мову програмування PHP [9] з архітектурою, основою на MVC (Model-View-Controller). Використання Laravel [10], одного з найпопулярніших PHP-фреймворків, дозволяє нам забезпечити швидку розробку, гнучкість, безпеку та підтримку стандартів. Вибір фреймворку Laravel для серверної частини нашої системи є обґрунтованим з кількох причин:

- ефективність і продуктивність: Laravel є одним з найпопулярніших PHP-фреймворків завдяки своїй продуктивності та ефективності. Він пропонує оптимізований код, швидке виконання та надійність, що дозволяє створювати високоякісні веб-додатки з низьким рівнем навантаження на сервер;
- зручність у використанні: Laravel пропонує простий та зрозумілий синтаксис, що робить його дружнім для розробників. Він має чисту архітектуру та ряд потужних функцій, таких як маршрутизація, контролери, моделі, міграції, що полегшує розробку та підтримку проекту;
- розширюваність: Laravel надає широкий спектр розширень та пакетів, що допомагають розширити функціональність вашого додатка. Це дозволяє легко і швидко додавати нові функції, модулі та інструменти до вашої системи;
- спільнота та документація: Laravel має активну спільноту розробників, що надає підтримку, рекомендації та готові рішення для багатьох завдань. Крім того, Laravel має добре написану документацію, яка спрощує процес вивчення та використання фреймворку.

Для полегшення розробки та локального тестування було вирішено використовувати OpenServer [11] - інтегроване розробницьке середовище, яке включає в себе Apache2 та додаткові компоненти для розробки веб-додатків. OpenServer дозволяє зручно налаштовувати та керувати веб-сервером, а також надає засоби для роботи з базами даних та іншими інструментами для розробників.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	19
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Для розробки клієнтської частини нашого веб-застосунку ми використовуємо бібліотеку React JS [12], яка базується на мові програмування JavaScript [13]. JavaScript є широко використовуваною мовою програмування, яка використовується для реалізації динамічної та інтерактивної поведінки веб-застосунків. React JS є потужним інструментом для створення інтерактивних та ефективних користувацьких інтерфейсів. Використання React JS дозволяє нам створити SPA (односторінкову програму), яка забезпечує швидке завантаження даних і миттєве оновлення сторінки без перезавантаження.

Для забезпечення зручного та привабливого дизайну ми використовуємо css-фреймворк, зокрема Material UI [14], який надає нам широкі можливості стилізації та компоненти згідно з сучасними дизайн-стандартами. Він надає готовий набір стильних компонентів, які мають сучасний вигляд та сприяють створенню консистентного та привабливого користувацького інтерфейсу. Також надає можливість створювати адаптивні веб-додатки, які адаптуються до різних розмірів екрану. Це особливо важливо для забезпечення коректного відображення та взаємодії з нашим додатком на різних пристроях, включаючи комп'ютери, планшети та мобільні пристрої.

Розробка проекту виконується з використанням інструменту PhpStorm. Цей IDE був обраний з декількох причин:

- інтуїтивний інтерфейс та зручна навігація;
- потужне середовище розробки з широкою підтримкою PHP та інших технологій;
- розширені можливості налагодження коду;
- інтеграція з іншими популярними інструментами розробки;
- можливість розширення та налаштування IDE під свої потреби.

Отож, використання Laravel з React [15] дозволить нам створити сучасний та потужний стек технологій для розробки веб-додатків. В результаті, обрані інструментальні засоби та описані вимоги забезпечують ефективну розробку, зручну роботу з кодом, швидке завантаження даних та створення привабливого

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		20

користувачького інтерфейсу. Вибір цих інструментів допомагає нам реалізувати наш проект з високою якістю та задоволенням вимог користувачів.

Висновки до першого розділу

У даному розділі був проведений аналіз напрямків використання інформаційних технологій для розробки системи для дистанційної форми навчання. Була поставлена задача, проведений аналіз аналогів програмного продукту та обрана архітектура системи для дистанційної форми навчання.

Під час аналізу аналогів програмного продукту було виявлено ряд подібних систем, що використовуються для дистанційного навчання. Це дало змогу виявити недоліки та переваги існуючих рішень і врахувати їх під час розробки нової системи.

При виборі архітектури системи для дистанційної форми навчання були враховані основні вимоги. Обрана архітектура має забезпечити ефективну організацію навчального процесу та зручний інтерфейс для користувачів. Для розробки системи були обрані певні інструментальні засоби, які відповідають вимогам проекту.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ДИСТАНЦІЙНОЇ ФОРМИ НАВЧАННЯ

2.1 Визначення варіантів використання і структури додатку

Розробка будь-якої інформаційної системи починається з визначення функціональних та нефункціональних вимог. Функціональні вимоги описують, які можливості має мати система і як вона має працювати, тоді як нефункціональні вимоги встановлюють якісні характеристики системи, такі як продуктивність, сприйняття та системні вимоги.

Функціональні вимоги:

- в системі передбачено можливість авторизації користувачів, а процес реєстрації здійснюється адміністратором, який також може встановлювати ролі для облікових записів, що створюються. Ці ролі включають учня та вчителя;
- валідація введених даних;
- створення модуля для проходження тестів;
- система повинна надавати користувачеві підходящий засіб перегляду потрібної інформації;
- вимоги до звітності;
- можливість збереження інформації: система повинна зберігати потрібну інформацію в базу даних.

Нефункціональні вимоги:

1. Сприйняття

- для звичайних користувачів, для того, щоб навчитися користуватися інформаційною системою, потрібно близько 1 години. Однак, досвідчені користувачі можуть оволодіти необхідними навичками протягом 15-30 хвилин;
- система повинна забезпечувати час відповіді на звичайні запити не більше 5 секунд, а на більш складні запити - не більше 10 секунд;

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		22

- інтерфейс повинен бути інтуїтивно зручним для користувача та не вимагати додаткової підготовки.

2. Продуктивність

- система повинна підтримувати мінімум 500 користувачів, що одночасно працюють, пов'язаних з спільною базою даних;
- швидке отримання результатів при здійсненні функцій програми.

Системні вимоги:

1. Вимоги до середовища виконання:

Система повинна задовольняти зазначеним вимогам на комп'ютері або смартфоні в наступній мінімальній комплектації:

- підтримка браузера *Google Chrome* версії 45+ або;
- підтримка браузера *Opera* версії 30+ або;
- підтримка браузера *Mozilla Firefox* версії 38+ або;
- підтримка браузера *Microsoft Edge* версії 12+ або;
- підтримка браузера *Safari* версії 9+

2. Вимоги до СУБД та доступу до даних:

У ядрі системи повинна бути представлена СУБД реляційного доступу.

Отож, після визначення всіх вимог, можливостей та обмежень системи, можна розробити діаграму варіантів використання [16], яка візуалізує взаємодію між акторами (користувачами) та системою у рамках конкретного функціонального сценарію або використання системи. Вона дозволяє ідентифікувати основні функції системи та деталізувати взаємодію між користувачами та системою у формі взаємодій (use cases).

Діаграма використання допомагає уточнити вимоги до системи, відображає, як система буде використовуватися в реальних сценаріях, та є важливим інструментом для розуміння функціональних вимог та взаємодії між компонентами системи.

Кольорами позначено частини роботи різних учасників команди.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	23
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

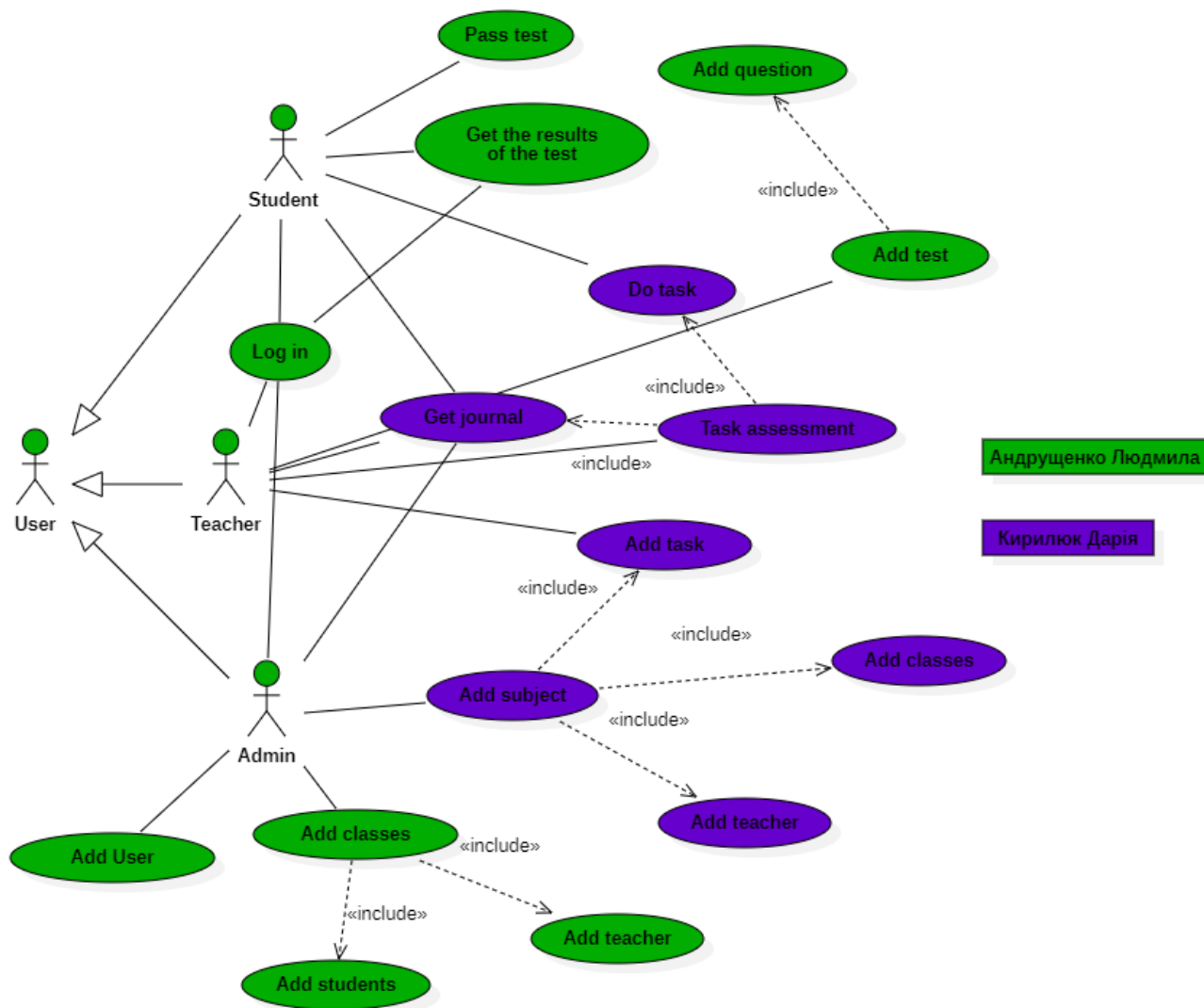


Рисунок 2.1 - Діаграма варіантів використання

Отож, з вище наведеного рисунку, можна зрозуміти, що система складається з трьох основних акторів: адміністратора, вчителя та учня. Кожен актор має свої унікальні функції та можливості в системі.

Звичайний користувач, який ще не автентифікувався в системі, має доступ лише до можливості автентифікації, тобто проходження процедури входу в систему для отримання ідентифікаційних прав та доступу до інших функцій.

Після успішного входу в систему, учень отримує доступ до наступних функцій: перегляд власних предметів, виконання завдань та проходження тестів. Після виконання завдань та тестів, учень зможе переглянути свій журнал, де будуть відображені результати його активностей та отримані оцінки.

В свою чергу, вчитель має розширені можливості в системі. Він може додавати завдання до предметів, які він викладає, перевіряти виконання цих завдань учнями та встановлювати оцінки за їх виконання. Отримані оцінки автоматично заносяться в журнал, що спрощує процес оцінювання та збереження відповідних даних. У разі якщо вчитель призначений класним керівником, він також має доступ до редагування класу. Це означає, що він може змінювати кількість учнів, додавати або видаляти їх зі списку класу, вносити необхідні зміни до інформації про класний колектив.

Адміністратор володіє повним контролем над системою. Він має можливість додавати нових користувачів до системи та керувати їх доступом до різних функцій і ресурсів. Адміністратор також відповідає за створення та редагування класів та предметів у системі, що дозволяє організувати навчальний процес. Крім цього, адміністратор має доступ до перегляду тестів, які були створені вчителями, та журналу класів. Це дозволяє адміністратору відстежувати активність учнів, їхні досягнення та оцінки, а також забезпечує контроль якості навчального процесу.

Після аналізу вимог до системи та створення діаграми використання, наступним кроком є побудова діаграми класів [17], яка буде розроблена за допомогою інструменту—StarUML [18]. Діаграма класів є важливим етапом, вона допомагає візуалізувати структуру системи, її основні сутності та зв'язки між ними.

Побудова діаграми класів передбачає ідентифікацію та визначення класів системи, їх атрибутів, методів та зв'язків. Кожен клас відображає певну сутність системи або компонент, який має свої властивості та поведінку. Основними елементами діаграми класів є класи, взаємозв'язки між класами (такі як асоціації, композиції, агрегації) та методи, які визначають операції, що можуть бути виконані над об'єктами цих класів.

Тому нижче наведена діаграма (рис.2.2) надає загальний огляд структури нашого проекту і відображає основні класи, їх атрибути та зв'язки між ними.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		25

Вона допомагає розуміти, як різні дані організовані в нашій системі та як вони взаємодіють між собою.

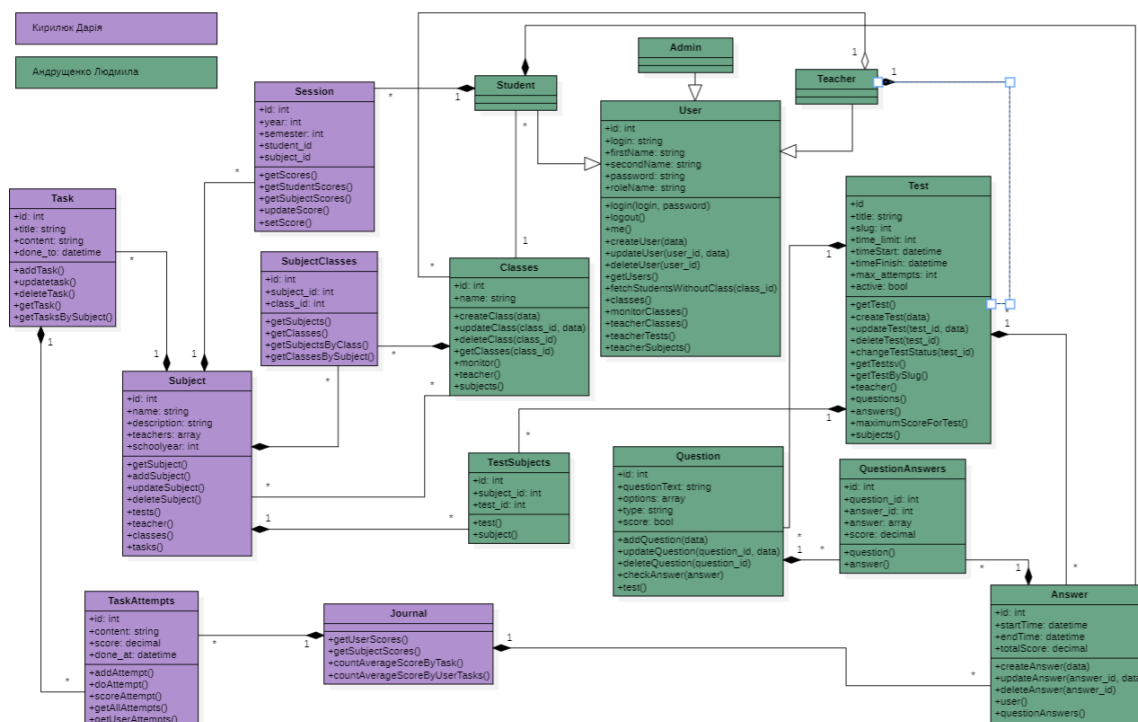


Рисунок 2.2 - Діаграма класів

Під час проектування діаграми для інформаційної системи з дистанційною формою навчання були визначені наступні класи, які описані нижче.

Клас User містить інформацію про зареєстрованого користувача, а саме таку як: ідентифікатор, логін, ім'я, прізвище, пароль та відповідну призначену роль. Клас User також надає ряд методів для взаємодії з користувачами у системі. Основні методи класу User включають:

- login(login,password) надає можливість здійснювати вхід в систему незареєстрованим користувачам, передаючи свій логін та пароль;
- logout() надає можливість здійснювати вихід з системи авторизованому користувачеві;
- me() призначений для отримання даних про поточного користувача;
- createUser(data) надає можливість адміністратору додавати нового користувача в систему, передаючи необхідні дані;
- updateUser(user_id, data) призначений для оновлення даних відповідного користувача;

- deleteUser(user_id) надає можливість видаляти потрібного користувача з системи;
- getUsers() призначений для отримання всіх користувачів системи;
- fetchStudentsWithoutClass(in class_id) надає можливість отримувати лише тих учнів, що не прив'язані до жодного класу;
- classes() надає можливість отримати клас, в якому навчається відповідний студент;
- monitorClasses() надає можливість отримати старосту класу;
- teacherClasses() надає можливість отримати класного керівника класу;
- teacherTests() повертає всі тести, які створив певний вчитель;
- teacherSubjects() надає можливість отримати всі предмети, які викладає поточний вчитель.

Клас Classes відповідає за модель представлення класів у інформаційній системі. Він дозволяє зберігати дані про класи, включаючи інформацію про старосту класу, класного керівника та студентів, які належать до цього класу. Його методи:

- createClass(data) дозволяє створювати новий клас та додавати до нього відповідних учнів та класного керівника;
- updateClass(class_id, data) надає можливість оновлювати дані обраного класу;
- deleteClass(class_id) дозволяє видаляти клас за вказаним ідентифікатором;
- getClasses(class_id) повертає всі класи, які наявні в системі;
- monitor() повертає старосту класу;
- teacher() надає можливість отримати класного керівника класу;
- subjects() повертає всі предмети, котрі вивчає відповідний клас.

Клас Test передбачає створення тесту для перевірки знань учнів з відповідного предмету. Він дозволяє зберігати різноманітні дані, пов'язані з тестом, такі як його назва, опис, тривалість, дата відкриття та закриття тощо. Він забезпечує структуроване зберігання цих даних та дозволяє легко отримувати

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		27

доступ до них. В свою чергу містить відповідні методи:

- `getTest(test_id)` надає можливість отримати тест за відповідним ідентифікатором;
- `createTest(data)` дозволяє додавати новий тест в систему;
- `updateTest(test_id, data)` призначений для оновлення даних відповідного тесту, вказуючи його ідентифікатор та нові дані;
- `deleteTest(test_id)` дозволяє видалити тест з системи за вказаним ідентифікатором;
- `changeTestStatus(test_id)` надає можливість змінити статус тесту на протилежний, тобто, якщо тест був неактивним, метод допоможе його активувати;
- `getTests()` дає можливість отримати всі тести, які наявні в системі;
- `getTestBySlug(slug)` надає можливість отримати тест за потрібною назвою;
- `teacher()` повертає вчителя, котрий створив даний тест;
- `questions()` повертає усі запитання, які містить обраний тест;
- `answers()` надає можливість отримати усі відповіді студентів;
- `maximumScoreForTest()` обраховує максимальний бал, який можна отримати за тест по 12-бальній шкалі;
- `subjects()` повертає всі предмети, для яких призначений відповідний тест.

Клас Question представляє модель питання у нашій системі. Він дозволяє зберігати дані, пов'язані з питаннями, такі як текст питання, варіанти відповідей, тип запитання, оцінка тощо. Методи класу:

- `addQuestion(data)` надає можливість користувачеві додавати нове питання до поточного тесту;
- `updateQuestion(question_id, data)` дозволяє оновлювати дані питання за його ідентифікатором;
- `deleteQuestion(question_id)` надає можливість видалити питання з відповідного тесту;

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		28

- `checkAnswer(answer)` перевіряє відповіді учня та повертає отриманий бал, за відповідне питання;
- `test()` повертає тест, до якого належить вказане питання.

Клас *Answer* відповідає за зберігання та управління відповідями певного користувача до тесту. Клас дозволяє зберігати дані, пов'язані з відповідями, такі як початок тестування та закінчення, ідентифікатор тесту, ідентифікатор студента, загальна оцінка. Клас містить такі методи :

- `createAnswer(data)` після того, як користувач завершив тестування, даний метод зберігає його відповідь у базі даних;
- `updateScoreForQuestionAnswer(answer_id, score)` надає можливість вчителю оновити бал за відповідь на питання. Він дозволяє вчителю встановити новий бал за певну відповідь, і це може бути використано, наприклад, для коригування балів після перегляду та оцінювання відповідей студентів на питання.
- `deleteAnswer(answer_id)` надає можливість, вчителю, за потреби, видаляти відповідь учня;
- `test()` повертає самий тест, до якого належить дана відповідь;
- `user()` призначений для отримання інформації про учня, пов'язаного з відповіддю. Він надає можливість отримати об'єкт користувача (учня), який зв'язаний з відповіддю на тест;
- `questionAnswers()` дозволяє отримати всі відповіді учня на питання, пов'язані з тестом. Він надає можливість отримати список об'єктів відповідей, які були надані учнем для кожного питання в рамках тесту.

Клас *QuestionAnswer* представляє модель відповіді учня на питання у системі. Він дозволяє зберігати дані, пов'язані з відповідями користувачів на питання. Це може включати ідентифікатор питання, ідентифікатор відповіді, текст відповіді та оцінка за відповідне питання. Методи :

- `question()` призначений для отримання самого питання до якого відноситься дана відповідь;

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	29
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

- answer() дозволяє отримати загальну відповідь, до якої відноситься дана відповідь на питання.

Клас TestSubject призначений для реалізації зв'язку багато до багатьох між класом Test та Subject. За допомогою нього, тест зможе призначатися для кількох предметів, а також, відповідно, предмети можуть містити в собі декілька тестів на перевірку знань учня. Методи, котрі наявні в тесті:

- test() дозволяє отримати об'єкт тесту, пов'язаного з об'єктом TestSubject. Він повертає тест, який призначений до певного предмету.
- subject() дозволяє отримати об'єкт предмета, пов'язаного з об'єктом TestSubject, повертає предмет, до якого призначений певний тест.

2.2 Розробка бази даних системи

Для ефективної роботи інформаційної системи необхідна добре структурована база даних, яка зберігатиме інформацію про користувачів, класи, предмети, тести та інші важливі дані. Важливо правильно спроектувати структуру бази даних, визначити відповідні типи даних для кожного атрибута і належним чином налаштувати зв'язки між таблицями, щоб забезпечити цілісність даних та ефективну роботу системи.

Оскільки розроблювальна платформа має значну кількість компонентів, вона також містить в собі різні таблиці в базі даних. В наступній таблиці та на рисунку представлені основні таблиці, які складають нашу БД. Ці таблиці відображають важливі аспекти системи та забезпечують збереження, організацію та доступ до даних.

Таблиця 2.1

Інформація про таблиці бази даних

Назва таблиці	Призначення
users	Для зберігання інформації про користувачів системи
classes	Для зберігання інформації про класи
tests	Призначена для зберігання тестів

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	30
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

questions	Для зберігання інформації про питання до відповідних тестів
answers	Для зберігання відповідей учнів
question_answers	Для збереження відповіді учня на конкретне питання тесту
test_subjects	Для реалізації зв'язку багато до багатьох tests та subjects

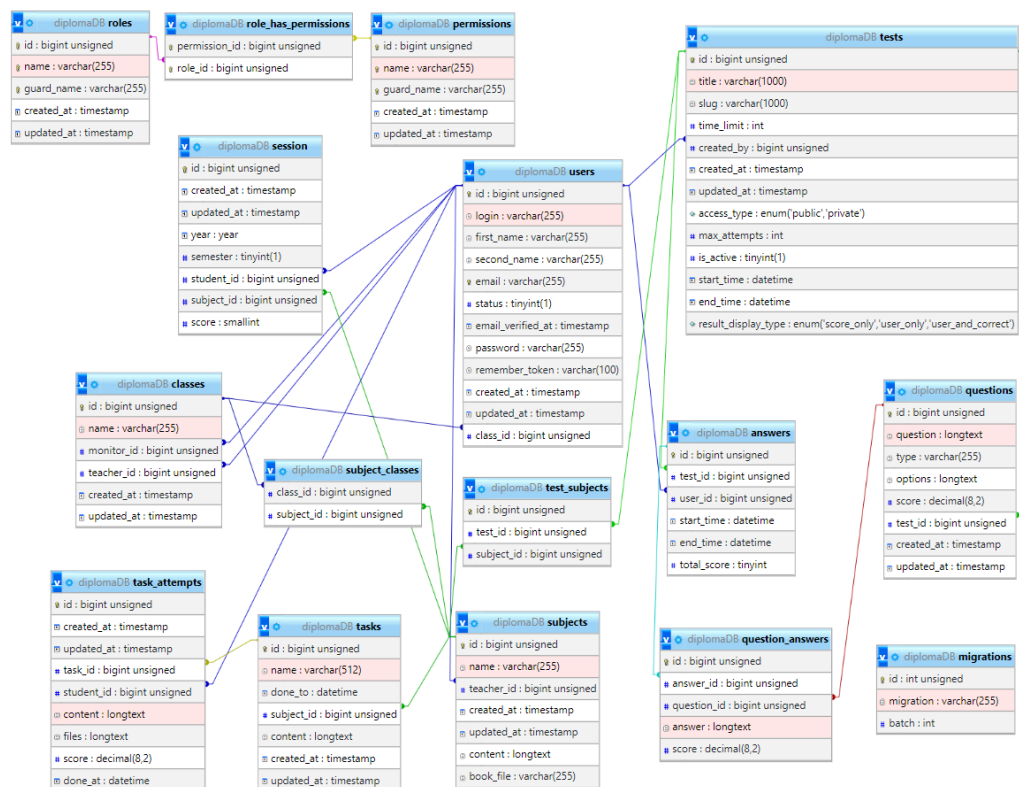


Рисунок 2.3 - Діаграма бази даних

Нижче наведений опис кожної з наведених таблиць бази даних (див. табл. 2.2 – 2.8)

Таблиця 2.2

Опис полів таблиці «users»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор користувача
login	varchar(255)	-	-	Логін користувачів

first_name	varchar(255)	-	-	Ім'я користувача
second_name	varchar(255)	-	-	Прізвище користувача
status	tinyint(1)	-	-	Статус, він відображає чи здійснювався вхід в систему після реєстрації адміністратором.
password	varchar(255)	-	-	Пароль користувача
created_at	timestamp	-	-	Це поле відображає дату і час створення запису в таблиці
updated_at	timestamp	-	-	Це поле відображає дату і час останнього оновлення запису в таблиці
class_id	bigint	-	+	Ідентифікатор класу, в якому навчається учень

Таблиця 2.3

Опис полів таблиці «classes»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор класу
name	varchar(255)	-	-	Назва класу
monitor_id	varchar(255)	-	-	Ідентифікатор учня, котрий є старостою класу
teacher_id	varchar(255)	-	-	Ідентифікатор вчителя, котрий є керівником класу
created_at	timestamp	-	-	Це поле відображає дату і час створення запису в таблиці
updated_at	timestamp	-	-	Це поле відображає дату і час останнього оновлення запису в таблиці

Опис полів таблиці «tests»

Назва	Тип даних	РК	FK	Опис поля
id	bigint	+	-	Ідентифікатор класу
title	varchar(255)	-	-	Назва класу
slug	varchar(255)	-	-	Поле для збереження “дружнього URL”
time_limit	varchar(255)	-	-	Максимальний час виконання тесту
created_by	timestamp	-	+	Ідентифікатор вчителя, який створив даний тест
max_attempts	int	-	-	Максимальна кількість спроб
is_active	tinyint(1)	-	-	Поле, яке призначене для активації/деактивації тесту
start_date	datetime	-	-	Дата відкриття тестування
end_date	datetime	-	-	Дата закриття тестування
result_display_type	enum('score_only', 'user_only', 'user_and_correct')	-	-	Зберігає в якому форматі показувати результат учневі
created_at	timestamp	-	-	Це поле відображає дату і час створення запису в таблиці
updated_at	timestamp	-	-	Це поле відображає дату і час останнього оновлення запису в таблиці

Опис полів таблиці «questions»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор питання
text	longtext	-	-	Текст питання
type	varchar(255)	-	-	Тип питання
options	longtext	-	-	Призначений для збереження варіантів відповідей
score	decimal(8,2)	-	-	Максимальна оцінка за правильну відповідь
test_id	bigint	-	+	Ідентифікатор тесту, до якого відноситься питання
created_at	timestamp	-	-	Це поле відображає дату і час створення запису в таблиці
updated_at	timestamp	-	-	Це поле відображає дату і час останнього оновлення запису в таблиці

Таблиця 2.6

Опис полів таблиці «answers»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор відповіді учня
test_id	bigint	-	+	Ідентифікатор тесту
user_id	bigint	-	+	Ідентифікатор учня
start_time	datetime	-	-	Час, коли учень почав тестування
end_time	datetime	-	-	Час, закінчення тестування
total_score	tinyint	-	-	Загальна оцінка за тест

Опис полів таблиці «question_answers»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор таблиці
answer_id	bigint	-	+	Ідентифікатор відповіді учня
question_id	bigint	-	+	Ідентифікатор запитання
answer	longtext	-	-	Відповідь учня
score	decimal(8,2)	-	-	Оцінка за питання

Таблиця 2.8

Опис полів таблиці «test_subjects»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Ідентифікатор таблиці
test_id	bigint	-	+	Ідентифікатор тесту
subject_id	bigint	-	+	Ідентифікатор предмету

База даних, описана вище, включає набір таблиць, які спеціально створені для керування інформацією про користувачів, класи, тести, питання, відповіді, зв'язки між питаннями та відповідями, а також предмети. Використання такої бази даних дозволяє зручно та ефективно організувати дані, забезпечуючи легкий доступ до них, збереження цілісності та зв'язків між різними сутностями.

2.3 Проектування та реалізація алгоритмів роботи додатку

На даному етапі розробки наведемо загальний алгоритм роботи програми, аби краще продемонструвати це, потрібно побудувати діаграму активності [19].

Діаграма активностей описує динамічні аспекти поведінки системи у вигляді блок-схеми, яка відображає бізнес-процеси, логіку процедур і потоки

робіт — переходи від однієї діяльності до іншої. По суті, ми маємо алгоритм дій (логіку поведінки) системи або взаємодії кількох систем.

На рисунку 2.4 зображено загальну діаграму діяльності.

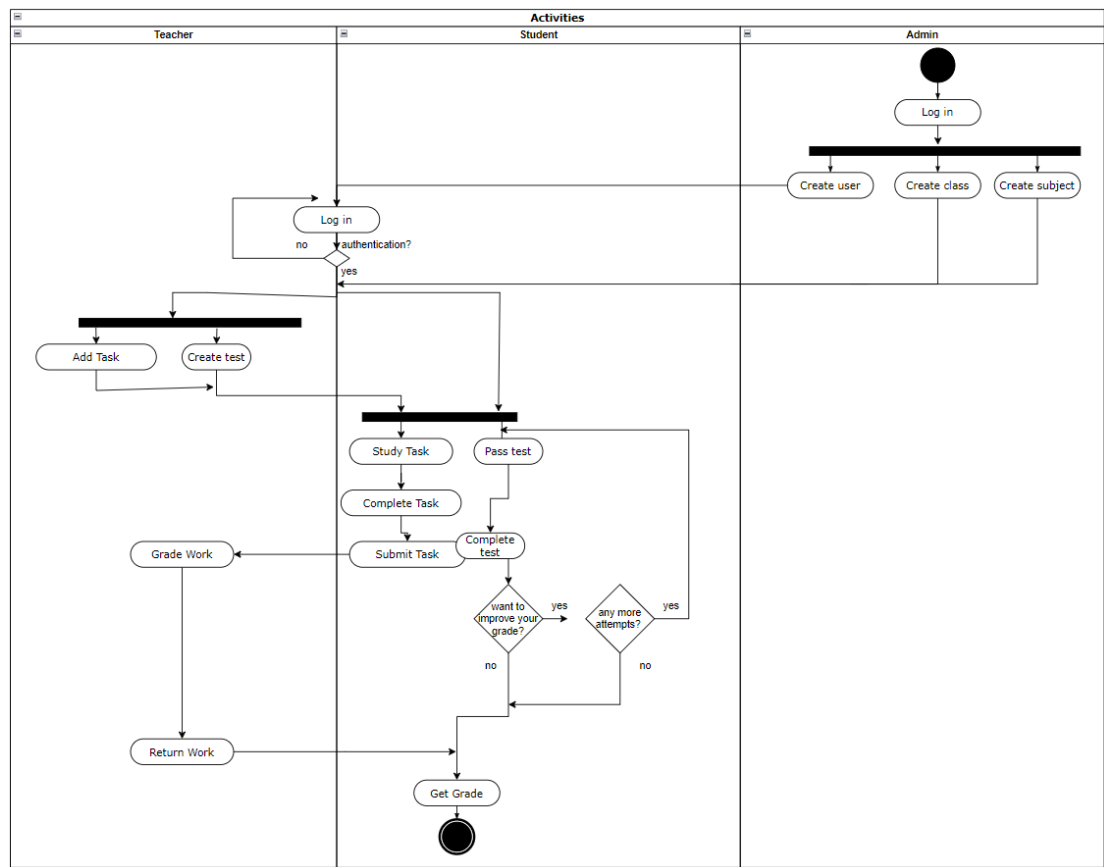


Рисунок 2.4 – Діаграма діяльності

Опис діаграми діяльності:

Адміністратор

- вхід (так – перехід до використання функцій системи, ні – завершення);
- дії:
 - створення користувача;
 - створення класу для здобувачів освіти;
 - створення предмету;

Студент

- вхід (так – перехід до використання функцій системи, ні – завершення);
- виконання завдання;

- надсилення виконаного завдання;
- проходження тесту;
- завершення проходження тесту;
- покращення результату;
 - якщо є додаткові спроби– повторне проходження тесту;
- перегляд оцінки за виконанні завдання.

Вчитель

- вхід (так – перехід до використання функцій системи, ні – завершення);
- додавання завдання;
- створення тесту;
- оцінювання роботи студента;
- повернення роботи з коментарями;

Наступна діаграма відображає етапи створення вчителем тесту для перевірки знань учнів та додавання до нього питань.

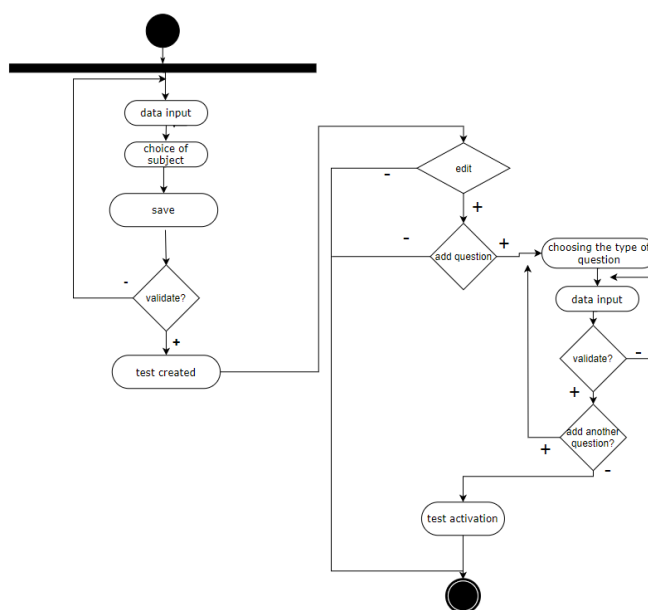


Рисунок 2.5 – Діаграма діяльності

Опис діаграми діяльності:

Для створення тесту, вчитель спочатку вводить необхідну інформацію, включаючи назву тесту, дату початку і закінчення, а також інші деталі. Потім вчитель має обрати відповідні предмети, до яких буде призначений тест.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	37
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Для подальшого редагування тесту, вчитель повинен перейти на сторінку створеного тесту, де може вносити потрібні зміни.

Щоб додати запитання до тесту, вчитель спочатку обирає тип питання, наприклад, одиночний вибір чи багатошаровий вибір, а потім вводить відповідні дані, включаючи текст питання, варіанти відповідей і т. д. При коректному введенні даних, питання зберігається у базі даних для подальшого використання.

Вже після виконання цих дій, вчитель може активувати тест, надаючи можливість учням проходити його і давати відповіді на запитання.

2.4 Реалізація функціоналу системи з підтримки дистанційної форми навчання для загальноосвітньої школи

В попередніх розділах було зазначено, що для розробки системи будуть використані: Laravel для розробки серверної частини та React для розробки клієнтської частини.

Одним із ключових аспектів успішної реалізації системи є правильна передача даних між клієнтом та сервером. Для цього був реалізований код у файлі `axios-client.js`, який забезпечує зручний і надійний механізм взаємодії між фронтендом та бекендом.

У наведеному фрагменті коду ми імпортуємо Axios [20] і створюємо екземпляр `axiosClient` за допомогою `axios.create()`. Ми вказуємо базову URL-адресу для всіх запитів до API серверної частини, використовуючи змінну середовища `VITE_API_BASE_URL`.

```
import axios from "axios";
const axiosClient = axios.create({
  baseURL: `${import.meta.env.VITE_API_BASE_URL}/api`
})
```

Далі встановлюємо перехоплювач запитів (interceptor), що додає авторизаційний токен до заголовків запитів. Ми отримуємо токен з локального сховища (localStorage) за допомогою `localStorage.getItem('ACCESS_TOKEN')` і присвоюємо його у заголовок Authorization запиту. Це дозволяє ідентифікувати та автентифікувати користувача на серверній стороні.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	38
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

```

axiosClient.interceptors.request.use((config) => {
  const token = localStorage.getItem('ACCESS_TOKEN')
  config.headers.Authorization = `Bearer ${token}`
  return config;
})

```

Щоб забезпечити безпеку і коректну обробку помилок авторизації на клієнтській стороні ми встановлюємо перехоплювач відповідей (interceptor) для обробки помилок. Якщо отримуємо відповідь з кодом статусу 401 (неавторизований доступ), то ми видаляємо авторизаційний токен з локального сховища (localStorage.removeItem('ACCESS_TOKEN')).

```

axiosClient.interceptors.response.use((response) => {
  return response;
}, (error) => {
  try {
    const {response} = error;
    if (response.status === 401) {
      localStorage.removeItem('ACCESS_TOKEN');
    }
  } catch (e) {
    console.error(e);
  }
  throw error;
})

```

Для зручного керування ролями та дозволами було використано популярний пакет Spatie/Laravel-Permission [21]. Він дозволяє визначати ролі користувачів, призначати їм дозволи та перевіряти доступ до певних функцій або ресурсів в залежності від ролей та дозволів. Тож розглянемо приклад створення ролі "admin" та призначення їй дозволів.

```

$admin = Role::create(['name' => 'admin']);
$arrayOfPermissionNames = [
    'show users',
    'create users',
    'delete users',
    'view classes',
    'create classes',
    'delete classes',
    'update classes',
];
$permissions = collect($arrayOfPermissionNames)->map(function
($permission) {return ['name' => $permission, 'guard_name' => 'web'];});
Permission::insert($permissions->toArray());
$admin->givePermissionTo(Permission::all

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	39
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Спочатку, за допомогою методу create створюємо нову роль з ім'ям admin, далі зберігаємо в масив всі дозволи які будуть призначені для даної ролі. Потім, за допомогою методу insert зберігаємо дозволи в БД, а далі метод givePermissionTo додає всі дозволи до відповідної ролі.

Також, варто зазначити, що при реалізації системи були використані бібліотеки Formik [22] і Yup [23] для зручного управління формами та їх валідації. Formik дозволяє спростити управління станом форми, а Yup забезпечує визначення правил валідації для даних введених у форму. Разом вони забезпечують зручний та надійний інтерфейс для введення та перевірки даних форми. Тож, розглянемо застосування даних бібліотек на прикладі створення користувача.

Спочатку визначаємо схему валідації, вона описує правила та обмеження для кожного поля форми, а також повідомлення про помилку, яке повинно відображатися у випадку невідповідності.

```
const validationSchema = Yup.object().shape({
  login: Yup.string().required('Поле логіну є обов'язковим для заповнення.').max(55, 'Логін занадто довгий').min(2, 'Логін занадто короткий').matches(/^S*$/, 'Ви не можете використовувати пробіли'),
  password: Yup.string().min(6, 'Пароль має містити принаймі 6 символів').max(100, 'Пароль занадто довгий').required('Поле паролю є обов'язковим для заповнення. '),
  first_name: Yup.string().min(2, 'Ім'я має містити принаймі 2 символи').max(55, 'Ім'я занадто довге').required('Поле імені є обов'язковим для заповнення').matches(/^[A-Za-z-яА-ЯЁёІіЇїЄєґґ]+$/, 'Для введення дозволенні лише літери'),
  second_name: Yup.string().min(2, 'Прізвище має містити принаймі 2 символи').max(55, 'Прізвище занадто довге').required('Поле прізвище є обов'язковим для заповнення').matches(/^[A-Za-z-яА-ЯЁёІіЇїЄєґґ]+$/, 'Для введення дозволенні лише літери'),
  role: Yup.string().required('Оберіть роль для користувача.'),});
```

Далі встановлюємо початкові значення (initial values), вони задають значення, які відображаються в полях форми при їх ініціалізації.

```
const initialValues = {
  login: '',
  password: '',
  role: '',
  first_name: '',
  second_name: '',};
```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		40

Наступним кроком, для того щоб управляти формою використовуємо `useFormik`, він повертає об'єкт, який містить різні властивості та методи, які можна використовувати для взаємодії з формою, такі як значення полів, обробники подій, стан валідації тощо.

```
const formik = useFormik({
  initialValues: user ? {...user} : {...initialValues},
  validationSchema,
  onSubmit: async (values, {}) => {
    if (!user) {
      const resultAction = await dispatch(createUser(values));
      ...
    } else {
      ...
      const resultAction = await dispatch(updateUser({id,
data}));
      ...
    }
  },
});
```

Далі розглянемо функцію відправлення запиту до серверу, на прикладі створення тесту.

```
export const createTest = createAsyncThunk(
  "tests/createTest",
  async (payload, {rejectWithValue}) => {
    try {
      const res = await axiosClient.post('/tests/create', payload);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з
сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);
```

У вище наведеному коді ми використовуємо функцію `createAsyncThunk` з `Redux Toolkit` [24] для створення асинхронної операції `createTest`. Вона приймає два параметри: перший параметр це префікс, який використовується для створення типів дій (у цьому випадку `"tests/createTest"`), а другий параметр це

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	41
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

асинхронна функція, яка містить логіку відправлення запиту до серверу. У середині цієї асинхронної функції виконується POST-запит на сервер з переданими даними payload. Якщо запит успішний, то повертається отримані дані (res.data). Якщо виникає помилка, перевіряється, чи є відповідь від сервера (error.response). Якщо немає відповіді, то викликається rejectWithValue з повідомленням про невдачу з'єднання з сервером.

У разі успішного запиту (createTest.fulfilled), стан Redux оновлюється за допомогою builder.addCase. Значення state.status встановлюється на 'succeeded', отримані дані (action.payload) додаються до масиву state.tests.

```
builder.addCase(createTest.fulfilled, (state, action) => {
  state.status = 'succeeded';
  state.tests.unshift(action.payload);
});
builder.addCase(createTest.rejected, (state, action) => {
  state.status = 'rejected';
  state.errors = action.payload;
});
```

Одним із ключових функціональностей нашої системи є тестування. Тож розглянемо метод store, який зберігає відповіді тестування в базу даних.

```
public function store(StoreAnswerRequest $request)
{
  $validated = $request->validated();
  ...
}
```

Вхідні дані отримуються через об'єкт \$request типу StoreAnswerRequest і перевіряються на валідність за допомогою методу \$request->validated(). Отримані дані зберігаються у змінній \$validated.

```
$studentAnswers = $validated['answers'];
unset($validated['answers']);
```

Змінна \$studentAnswers містить дані про відповіді студента, які були відправлені разом із запитом. За допомогою unset(\$validated['answers']) видаляються відповіді зі змінної \$validated, оскільки вони будуть збережені окремо.

```
$answerCreated = Answer::create($validated);
```

Створюється новий запис в таблиці Answer за допомогою методу Answer::create(\$validated). Отриманий об'єкт запису зберігається у змінну \$answerCreated.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	42
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

```

$totalScore = 0;
foreach ($studentAnswers as $questionId => $studentAnswer) {
    $question = Questions::where(['id' => $questionId, 'test_id'
=>
        $validated['test_id']])->first();
    if (!$question) {
        return response("Invalid question ID: \"\$questionId\"",
400);
    }
    $scoreForQuestion = $question->checkAnswer($studentAnswer);
    $data = [
        'question_id' => $questionId,
        'answer_id' => $answerCreated->id,
        'answer'      => is_array($studentAnswer)
?
        json_encode($studentAnswer) : $studentAnswer,
        'score' => $scoreForQuestion,
    ];
    $totalScore += $scoreForQuestion;
    QuestionAnswer::create($data);
}

```

Далі перебираємо кожну з відповідей учня на питання, якщо дане питання наявне в БД, перевіряємо відповідь учня, за допомогою методу `checkAnswer` (можна переглянути в додатках) та встановлюємо отриманий бал за питання. Створюється масив `$data`, який містить інформацію про запитання, відповідь студента, отриманий бал за запитання і на основі нього створюється новий запис в таблиці.

```

$test = Test::find($validated['test_id']);
$maximumScore = $test->maximumScoreForTest();
$answerCreated->total_score = 12 * ($totalScore / $maximumScore);
$answerCreated->save();
return response(new TestResource($test), 201);
}

```

Далі знаходимо тест, до якого належить відповідь та оновлюємо загальний бал отриманий студентом за тест.

Висновки до другого розділу

На поточному етапі виконання дипломної роботи був проведений аналіз і визначені вимоги до системи, яка розробляється. Також були встановлені мінімальні системні вимоги, необхідні для належної роботи додатку. З метою

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	43
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

детального уявлення про функціонал системи, була створена діаграма варіантів використання.

Додатково, був розроблений загальний алгоритм, за яким буде виконуватися робота веб-сайту. Цей алгоритм був відображений у вигляді діаграми діяльності. Для кращого розуміння та визначення взаємодії об'єктів системи для дистанційного навчання, була розроблена об'єктно-орієнтована модель.

Крім цього, у цьому розділі була розроблена структура бази даних.

У заключному пункті наведено докладний розгляд деяких аспектів реалізації: описано процес передачі даних між клієнтом та сервером, а також згадано про виконання валідації даних та інші подібні аспекти.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		44

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ

3.1 Порядок встановлення та налаштування параметрів системи

Створений веб-сайт є веб-орієнтованою системою підтримки дистанційної форми навчання для загальноосвітньої школи. Це означає, що він доступний через багато браузерів і призначений для використання на різних пристроях, таких як телефони, планшети та ноутбуки.

Для користування цим веб-сайтом потрібен пристрій, який відповідає певним мінімальним вимогам. Наприклад, операційна система повинна бути Windows 7 або новіша для ПК, або MacOS 10.11 або новіша для Mac-пристроїв. Процесор повинен мати тактову частоту від 1,3 до 3,8 ГГц, а ОЗУ - не менше 2 ГБ. Також можна використовувати пристрої на базі Android 6.0 або новішої версії.

Важливою перевагою цього розробленого додатку є те, що він не потребує спеціалізованого обладнання або додаткових налаштувань, окрім стандартних компонентів, які встановлені на більшості сучасних операційних систем. Це означає, що користувачам буде зручно користуватися сайтом без необхідності виконувати складні налаштування чи встановлювати спеціальне програмне забезпечення.

Загалом, цей додаток створений з урахуванням максимальної зручності користувачів. Він буде працювати на багатьох операційних системах і пристроях, що сприятиме позитивному досвіду користувачів.

Для встановлення та налаштування сервісу дистанційного навчання необхідно розгорнути всі компоненти системи та правильно налаштувати як кожен окремий елемент, так і загальну систему в цілому.

Для опису архітектури та процесу розгортання мобільного сервісу була підготовлена діаграма розгортання [25] (рис. 3.1), яка ілюструє взаємодію між різними компонентами системи та їх розміщення на серверах та мобільних

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	45
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

пристроях.

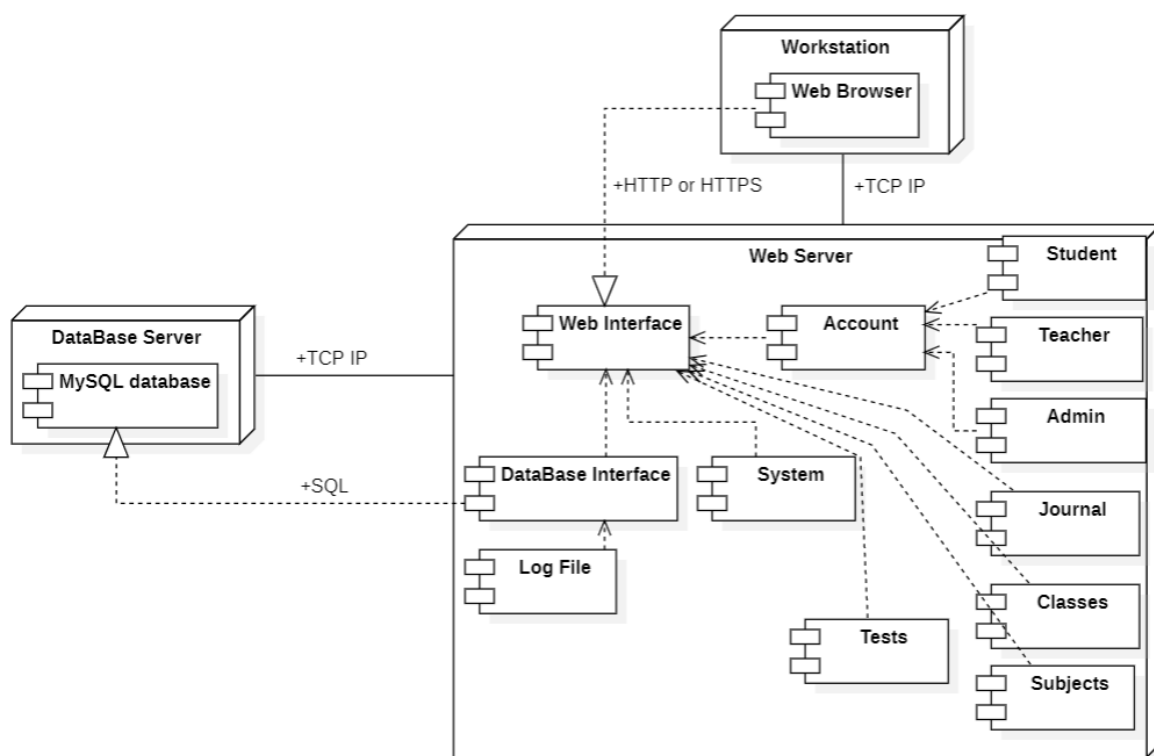


Рисунок 3.1 - Діаграма розгортання системи

На діаграмі розгортання відображаються компоненти та об'єкти, які працюють на обчислювальних вузлах під час роботи програми. Ця діаграма надає структурований огляд розміщення різних елементів системи. В контексті нашого проекту, діаграма розгортання має наступну структуру:

- **Web Interface** – це шаблони сайту, які використовуються в нашому проекті для взаємодії з користувачами системи через веб-браузер;
- **DataBase Interface** – цей компонент відповідає за взаємодію з базою даних, забезпечуючи доступ до неї та обробку запитів;
- **Log file** – файл, який використовується системним адміністратором для відстеження та налагодження роботи системи. Він містить інформацію про події та помилки, що виникають під час роботи системи.
- **Web Browser** – програмний продукт, встановлений на пристрої користувача, який використовується для взаємодії з системою. Він дозволяє користувачам отримувати доступ до функціоналу системи через веб-інтерфейс.

3.2 Структура інтерфейсу системи з підтримки дистанційної форми навчання для загальноосвітньої школи

Веб-застосунок системи надає можливість користувачам взаємодіяти з системою шляхом використання облікових записів. Для початку роботи в системі необхідно увійти, за допомогою введення логіну та паролю в спеціальні поля на сторінці входу. Спершу, увійдемо в обліковий запис адміна.

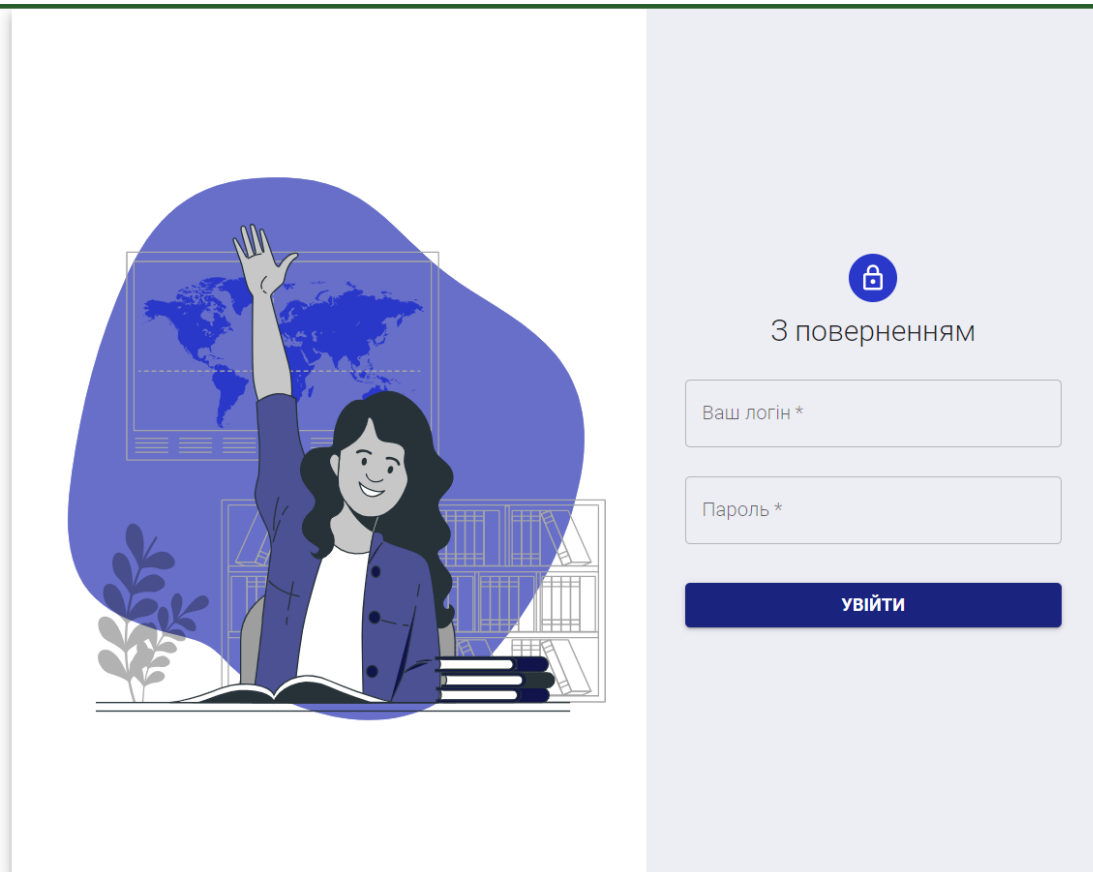


Рисунок 3.2 – Сторінка входу

Система перевіряє введені дані та здійснює перевірку на відповідність збереженим обліковим записам адміністраторів. Введені дані є вірними, тож тепер, ми можемо користуватися функціоналом системи. Зліва на веб-сторінці зображено меню сайту, за допомогою якого ми можемо легко переміщуватися по додатку.

Давайте перейдемо на сторінку користувачів, де відображаються всі користувачі, які наявні в системі.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	47
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

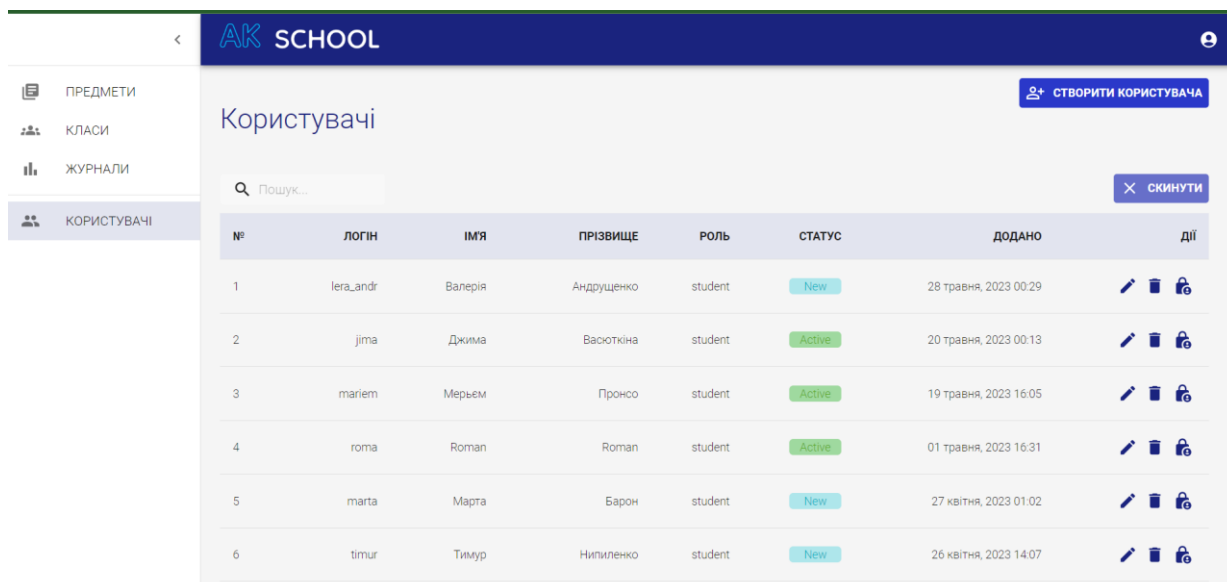


Рисунок 3.3 – Сторінка з відображенням користувачів

На сторінці користувачів ми маємо можливість здійснювати пошук і сортування за різними полями. Це дає нам зручну можливість швидко знайти конкретного користувача або відсортувати їх за певними критеріями.

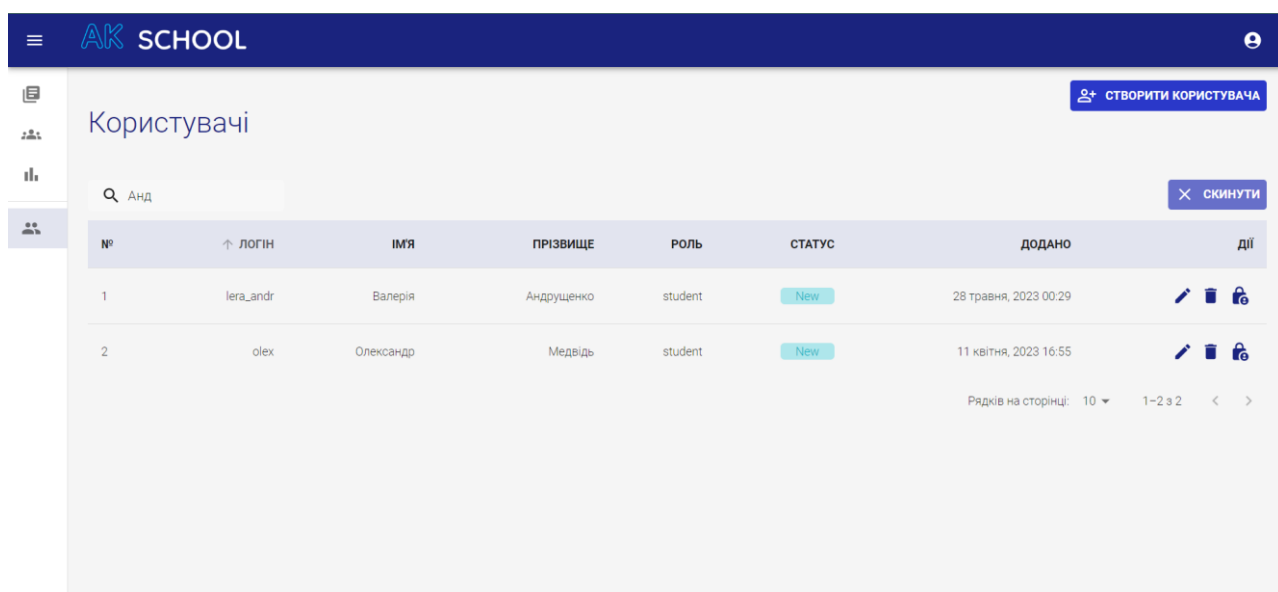


Рисунок 3.4 – Операції на сторінці користувачів

Також на сторінці користувачів адміністратор може переглядати профілі один за одним або використовувати функції пошуку і фільтрації для знаходження конкретного користувача за певними критеріями. При перегляді профілю користувача, адміністратор може побачити інформацію, таку як ім'я, контактні дані, роль в системі, статус облікового запису та інші важливі деталі.

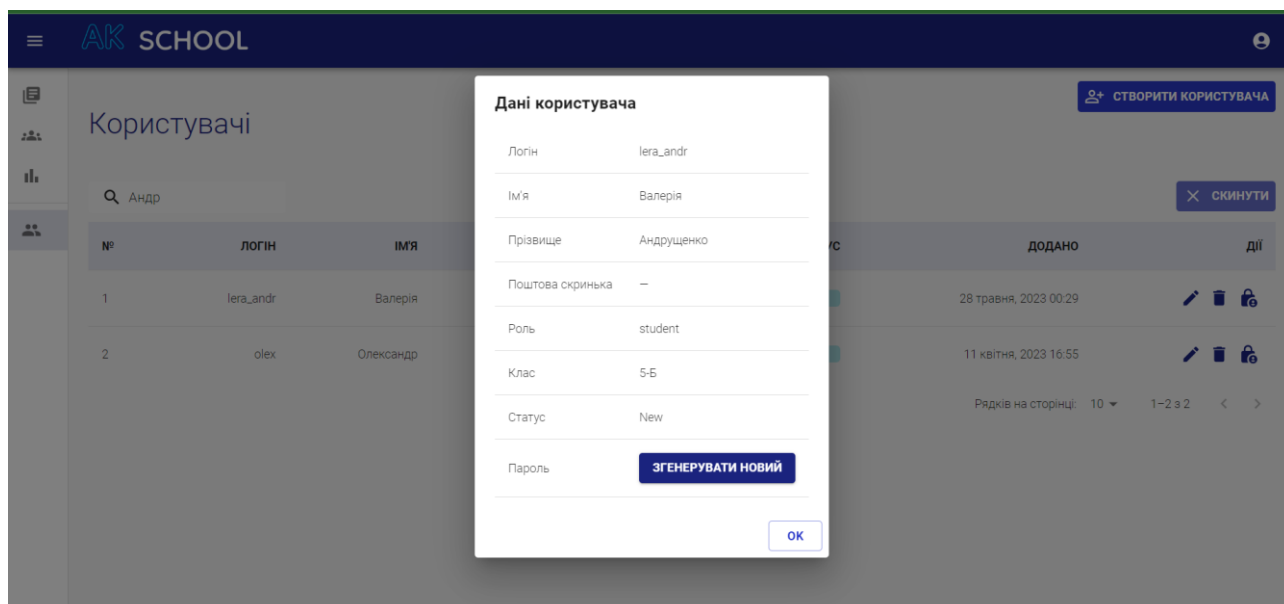


Рисунок 3.5 – Інформація про користувача

Враховуючи наші попередні знання, адміністратор має здатність створювати нові облікові записи користувачів у системі. Тепер ми спробуємо виконати цю операцію, заповнивши необхідні поля та згенерувавши пароль для створення нового користувача і натиснувши кнопку "Додати".

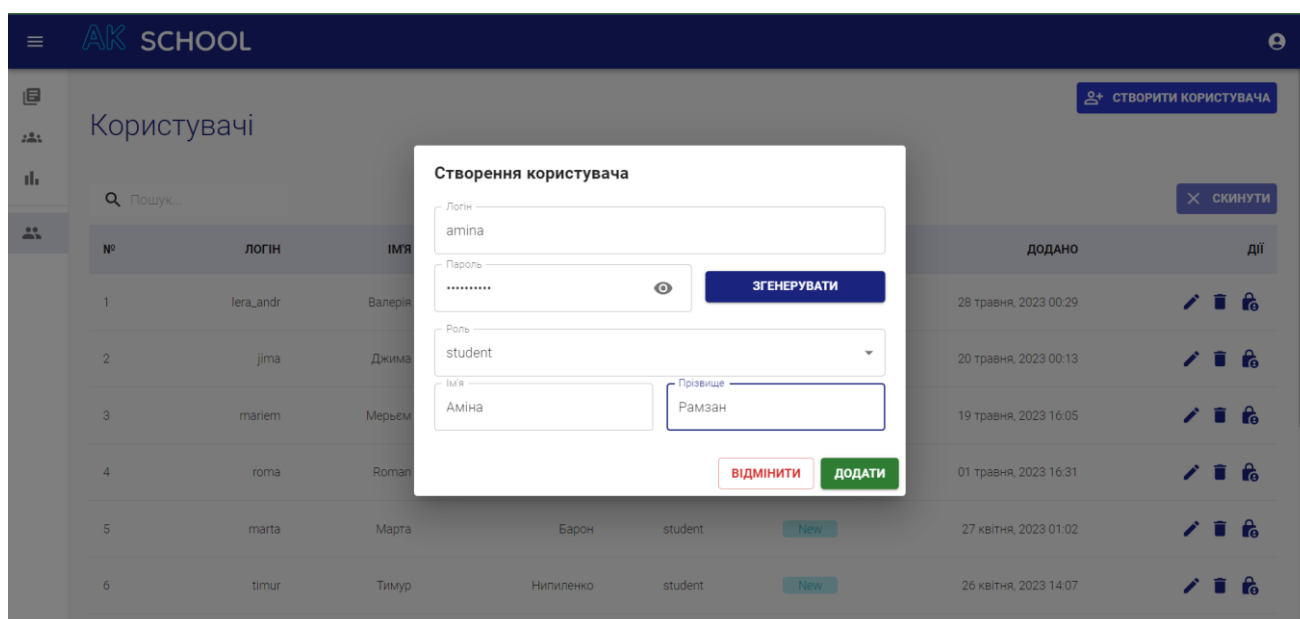


Рисунок 3.6 – Створення нового користувача

Після успішного виконання операції додавання користувача в систему, користувачу надсилається повідомлення про успішність дії. Система також надає можливість переглянути відразу створеного користувача у списку.

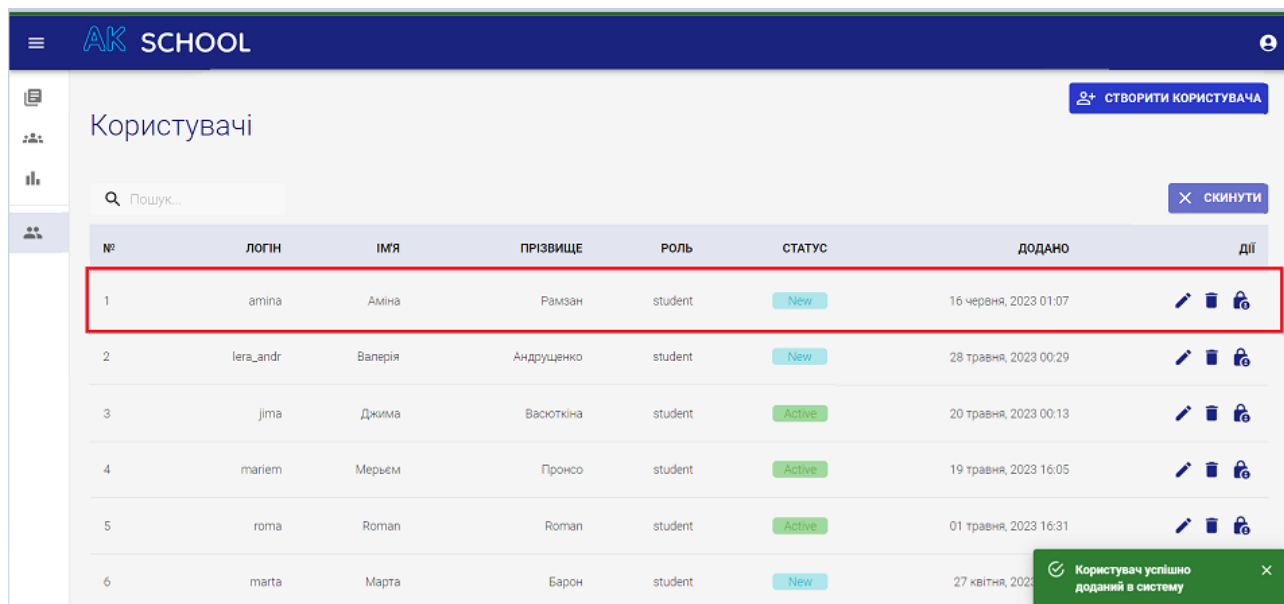


Рисунок 3.7 – Успішне виконання операції

Після успішного додавання декількох учнів до системи, ми маємо можливість об'єднати їх у клас. Для цього перейдемо на сторінку класів, де будуть відображені всі наявні класи у системі. На цій сторінці ми можемо знайти детальну інформацію про кожен клас, таку як: назва класу, класний керівник та список учнів, які входять до цього класу.

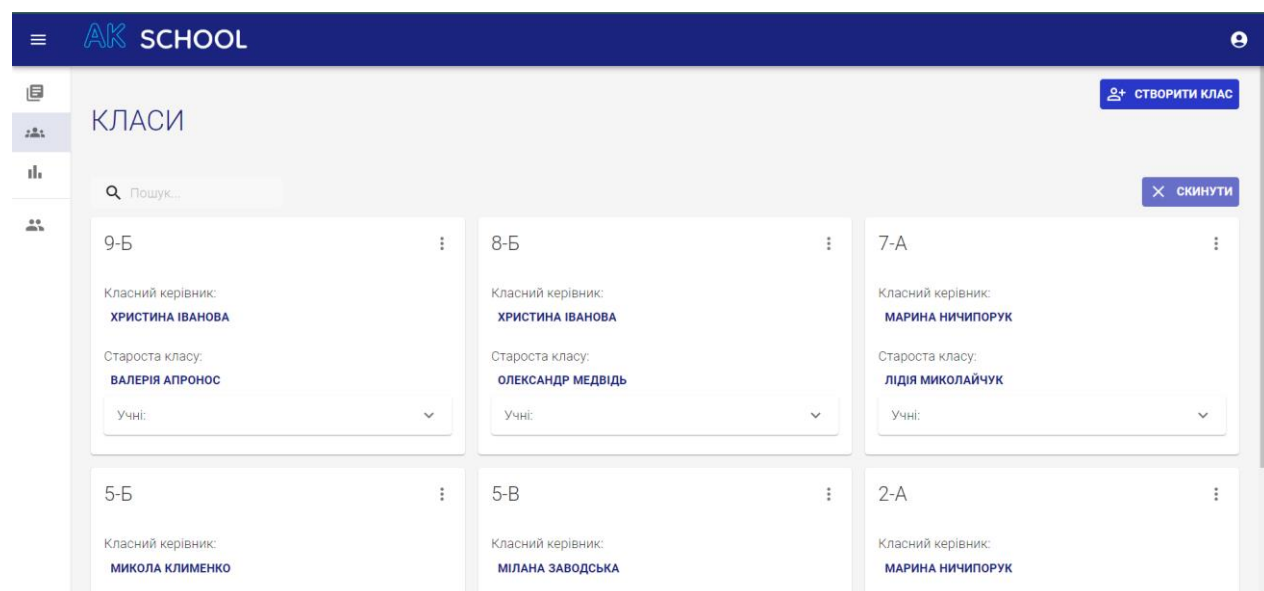


Рисунок 3.8 – Сторінка з відображенням класів системи

Тепер давайте створимо новий клас у системі. Під час створення класу, ми вводим назву класу, яка буде ідентифікувати його в системі. Потім ми можемо вибрати учнів, які будуть призначені до цього класу, зазначивши їх імена або вибравши їх зі списку користувачів. Крім того, ми можемо обрати старосту

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	50
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

класу, а також призначити класного керівника, який буде вести навчальний процес у класі.

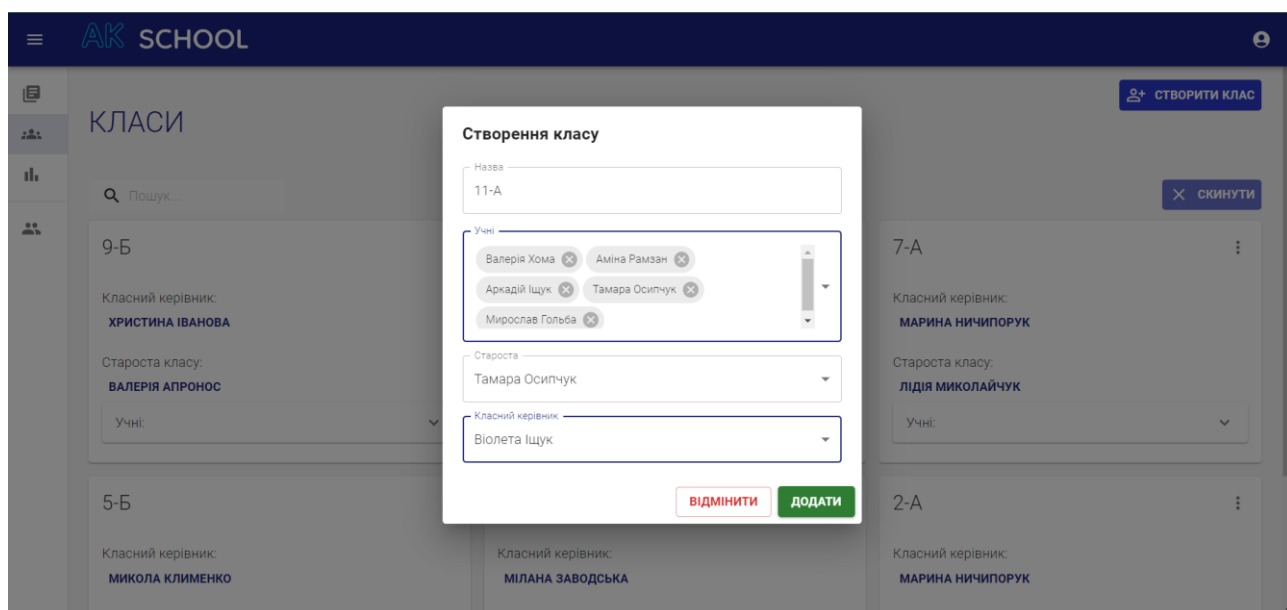


Рисунок 3.9 – Створення класу

Після успішного створення класу, система надсилає нам повідомлення, що операція пройшла успішно. Крім того, на сторінці класів відображається новостворений клас з відповідною інформацією.

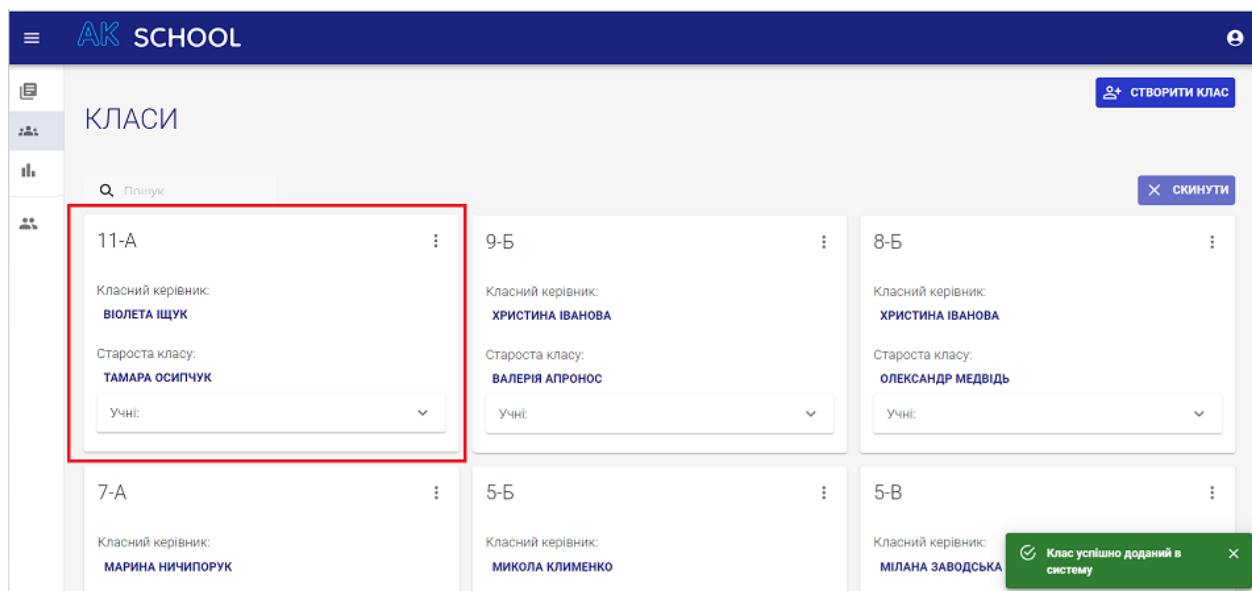


Рисунок 3.10 – Успішне створення класу

Зараз ми увійдемо в обліковий запис вчителя. Вчителю доступні різні пункти меню, які дозволяють йому здійснювати операції. На сторінці класів він може побачити список класів, в яких він є класним керівником або викладає певний предмет. Це дозволяє вчителю зручно відстежувати свої класи, знаходити

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	51
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

необхідну інформацію про них та виконувати потрібні дії, пов'язані з керуванням класами. На нижче наведеному зображенні(рис.3.11) розташовується сторінка класів.

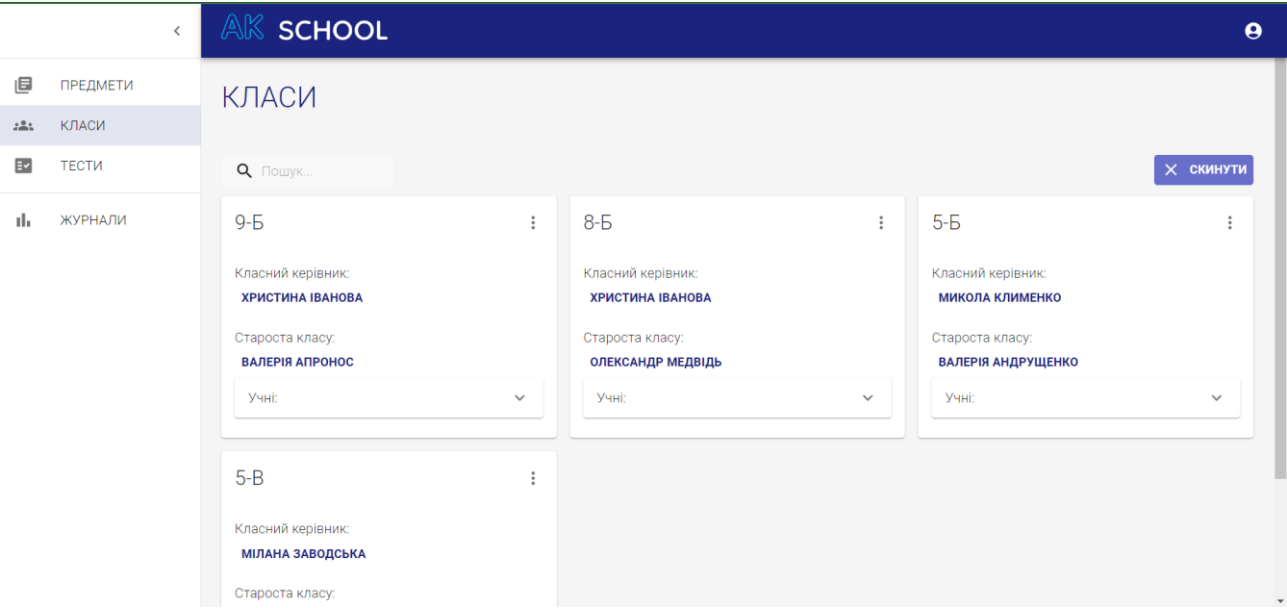


Рисунок 3.11 – Класи вчителя

На сторінці тестів вчителя він може переглянути всі тести, які він створив. Це дозволяє вчителю легко знайти тест, який йому потрібен, і переглянути його деталі.

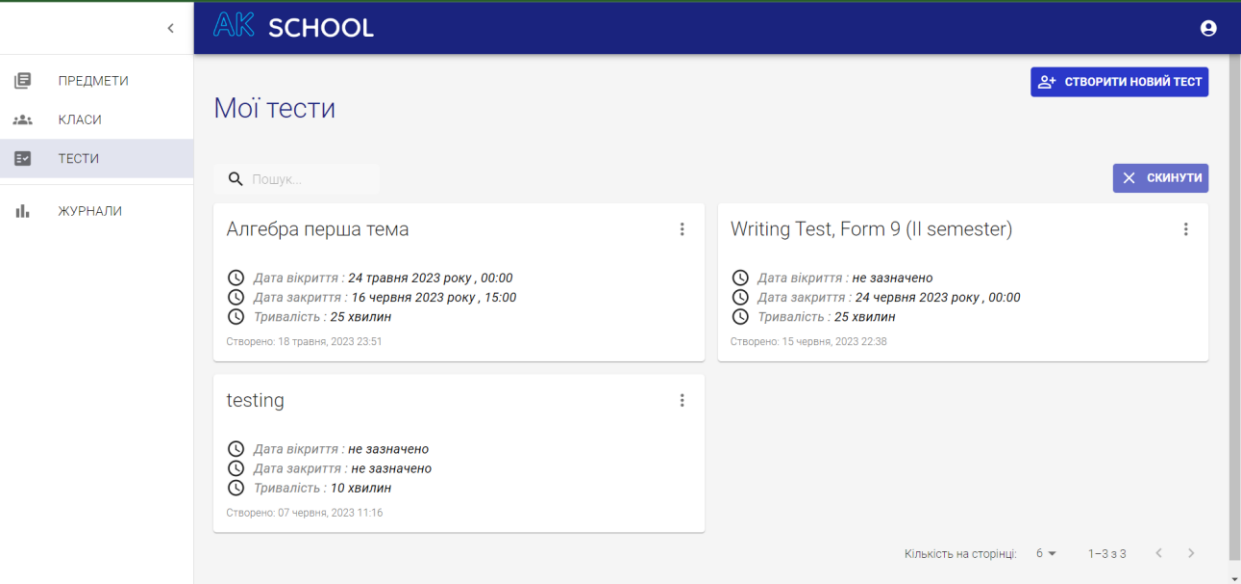


Рисунок 3.12 – Сторінка тестів

Давайте перейдемо до створення нового тесту. Спочатку виберемо назву тесту, яка відповідає його змісту та цілі. Далі оберемо предмети, до яких цей тест буде відноситися, що дозволяє зорієнтуватися в контексті його використання.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	52
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

При бажанні, ми можемо налаштувати додаткові параметри тесту, такі як тривалість, максимальна кількість спроб, дата початку та завершення.

Рисунок 3.13 – Створення тесту

Після успішного створення тесту, ми можемо перейти на його сторінку для керування ним. На цій сторінці будуть відображені всі додані питання, які входять до складу тесту, а також результати учнів, які проходили тестування. Крім того, на сторінці будуть доступні різноманітні кнопки для керування тестом. Наприклад, ми можемо додати питання різних типів, таких як питання з вибором правильного варіанту, питання з декількома варіантами відповідей, питання на встановлення відповідності або питання, що потребують короткої відповіді. Також ми маємо можливість активувати тестування, щоб дозволити учням проходити його. Крім того, на сторінці присутні кнопки для видалення тесту або редагування його змісту.

Рисунок 3.14 – Сторінка створеного тесту

Тепер, спробуємо додати декілька питань.

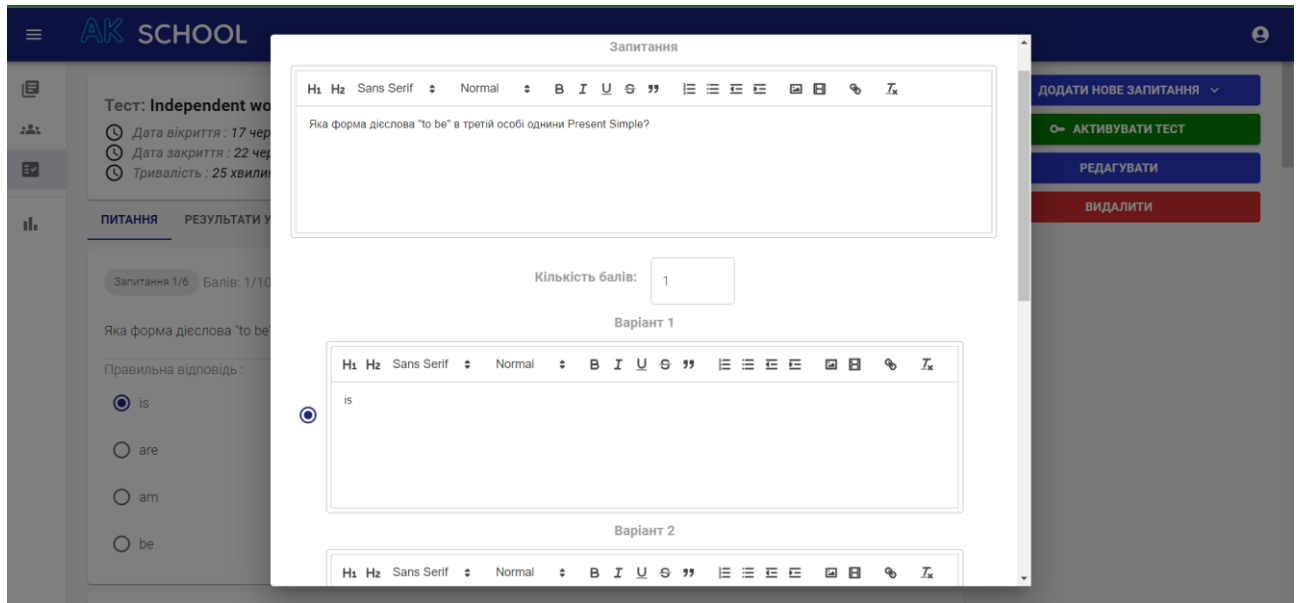


Рисунок 3.15 –Додавання питань

Після успішного додавання питань до тесту, у нас з'являється можливість активувати його. Активація тесту означає зробити його доступним для учнів, тобто відкритим для проходження. Після активації, учні зможуть перейти на сторінку тесту, відповісти на питання та подати свої відповіді.

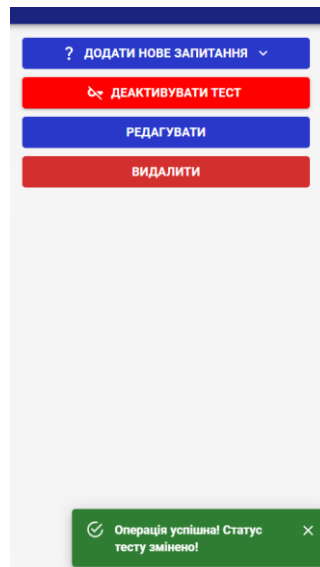


Рисунок 3.16 – Активація тесту

Перейдемо в обліковий запис учня, аби спробувати пройти тестування.

Після того, як ми увійшли в обліковий запис учня, ми маємо можливість пройти тестування. Для цього натискаємо кнопку "Розпочати тестування". Після натискання кнопки, система перенаправляє нас на сторінку з активним тестом.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	54
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

На цій сторінці ми можемо бачити питання, варіанти відповідей, за наявності часових рамок, таймера. Після того, як відповіли на всі питання, ми можемо натиснути кнопку "Завершити тестування" для подання відповідей.

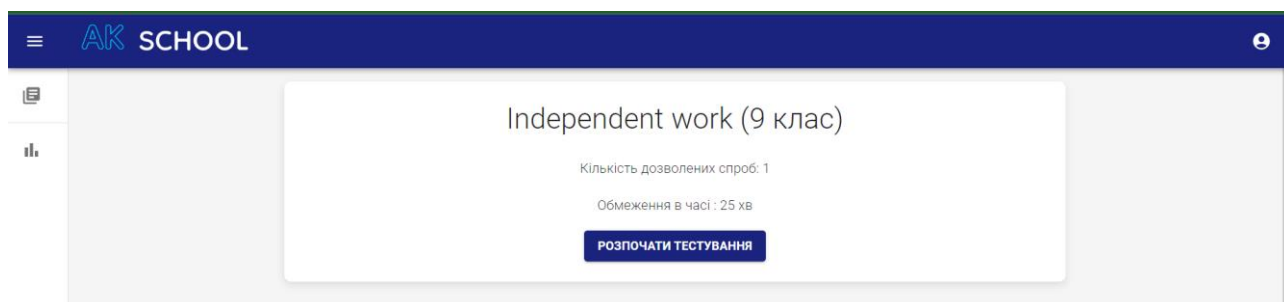


Рисунок 3.17 – Старт тестування

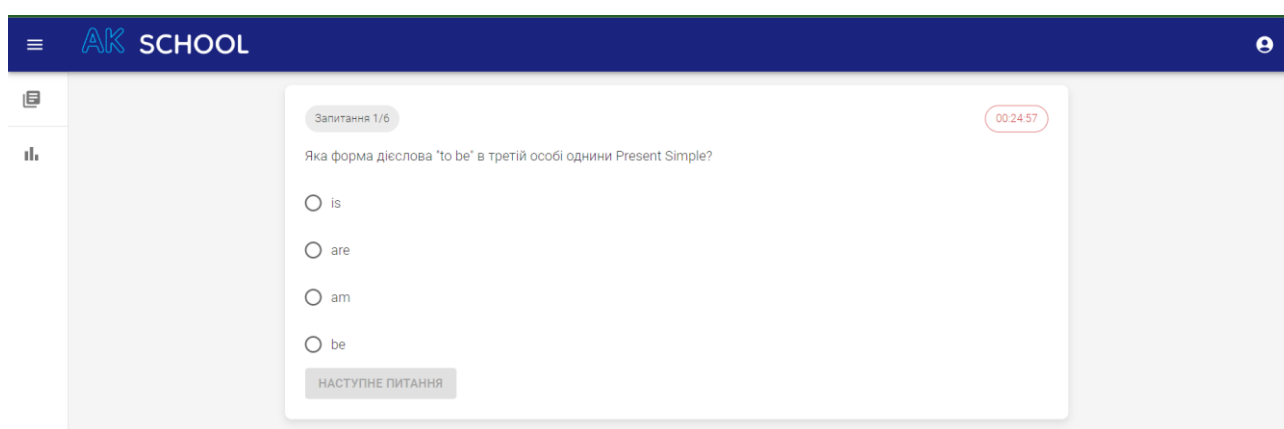


Рисунок 3.18 – Проходження тесту

Після натиснення кнопки "Завершити тестування", система обробляє відповіді учня і відображає його результат. На сторінці результатів ми можемо побачити оцінку, яку отримав учень за тест, а також деталізовану інформацію про його відповіді на кожне питання.

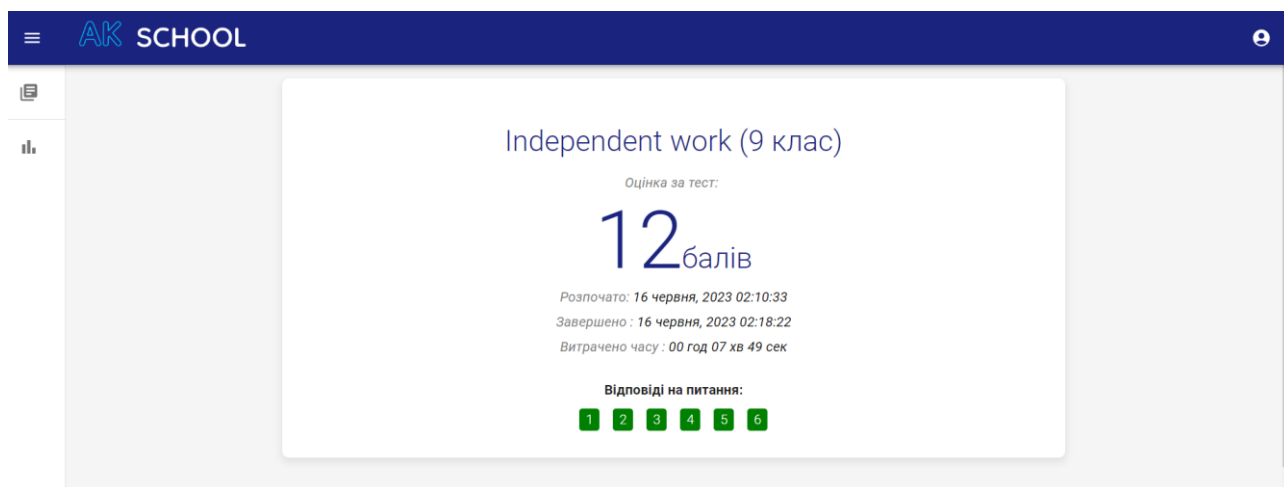


Рисунок 3.19 – Проходження тесту

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	55
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Після того, як учні завершують тестування, вчитель отримує можливість переглядати їхні результати. На сторінці результатів вчитель може побачити оцінки, які отримали учні за кожен тест, а також детальну інформацію про відповіді кожного учня.

Рисунок 3.20 – Результати учнів

3.3 Тестування роботи додатку

Один з важливих етапів в розробці програмного забезпечення - це процес тестування та виправлення помилок.

Початковий етап перевірок включає авторизацію на сторінці, де користувачу потрібно ввести свій логін та пароль (див. рис. 3.20). Всі поля, необхідні для авторизації або реєстрації, є обов'язковими, і будуть виводитись попередження у разі їхньої відсутності.

Рисунок 3.21 –Помилка при надісланні пустої форми

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	56
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

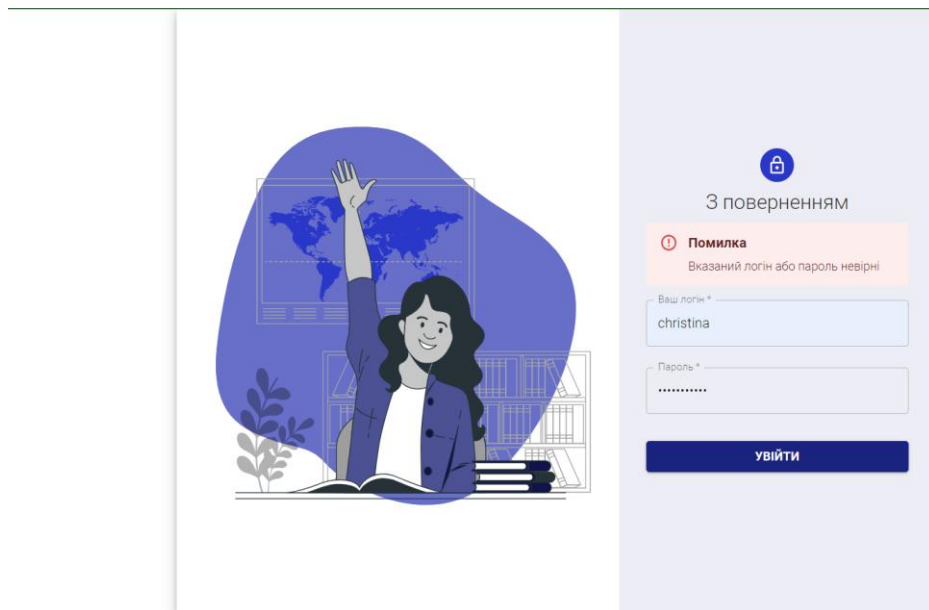


Рисунок 3.22 –Помилка при введенні некоректних даних

Якщо поля введення залишені порожніми або заповнені некоректними даними при відправленні форми, то система повідомляє про помилку. Вона виділяє червоним кольором поля з некоректними даними або виводить текстове повідомлення, яке пояснює, які поля не заповнені або містять невірні дані.

Це допомагає користувачеві відразу помітити помилки та виправити їх, забезпечуючи введення коректних даних в систему. Такі перевірки є важливим етапом для забезпечення цілісності та достовірності даних, що вводяться в систему.

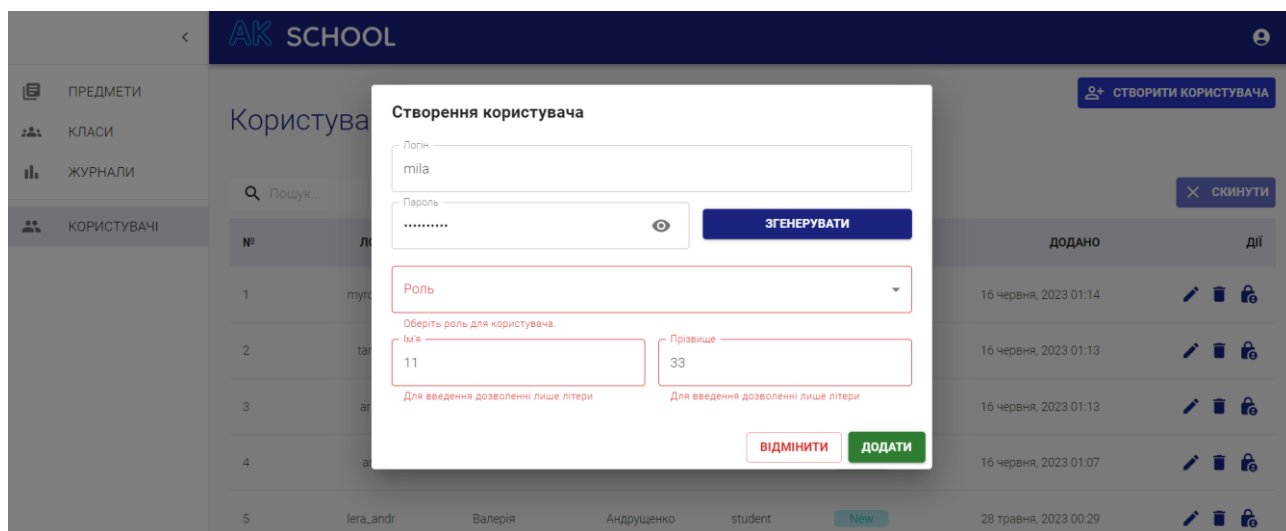


Рисунок 3.23 –Некоректне введення даних

Перевіримо реакцію системи на введення адреси, яка фактично не існує. Це важливий етап тестування, оскільки він дозволяє нам визначити, як система взаємодіє з некоректними або недоступними даними.

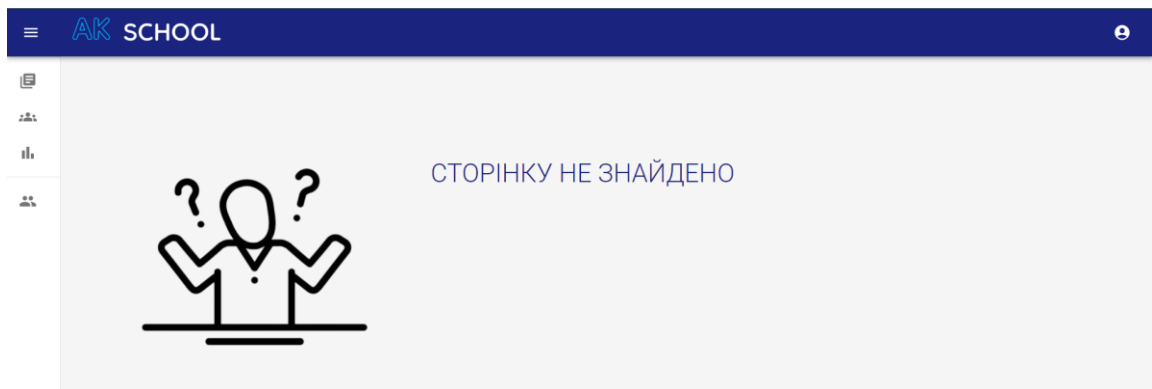


Рисунок 3.24 –Неіснуюча сторінка

Отже, система ефективно обробляє некоректну адресу та забезпечує користувачеві відповідну зворотну інформацію.

Також тестування адаптивності сайту є важливим етапом процесу тестування. Адаптивний дизайн означає, що сайт здатний адаптуватися до різних розмірів екранів і пристроїв, забезпечуючи оптимальний вигляд та функціональність для користувачів незалежно від того, з якого пристрою вони отримують доступ до сайту.

Тому, спочатку ми проведемо тестування адаптивності на планшеті.

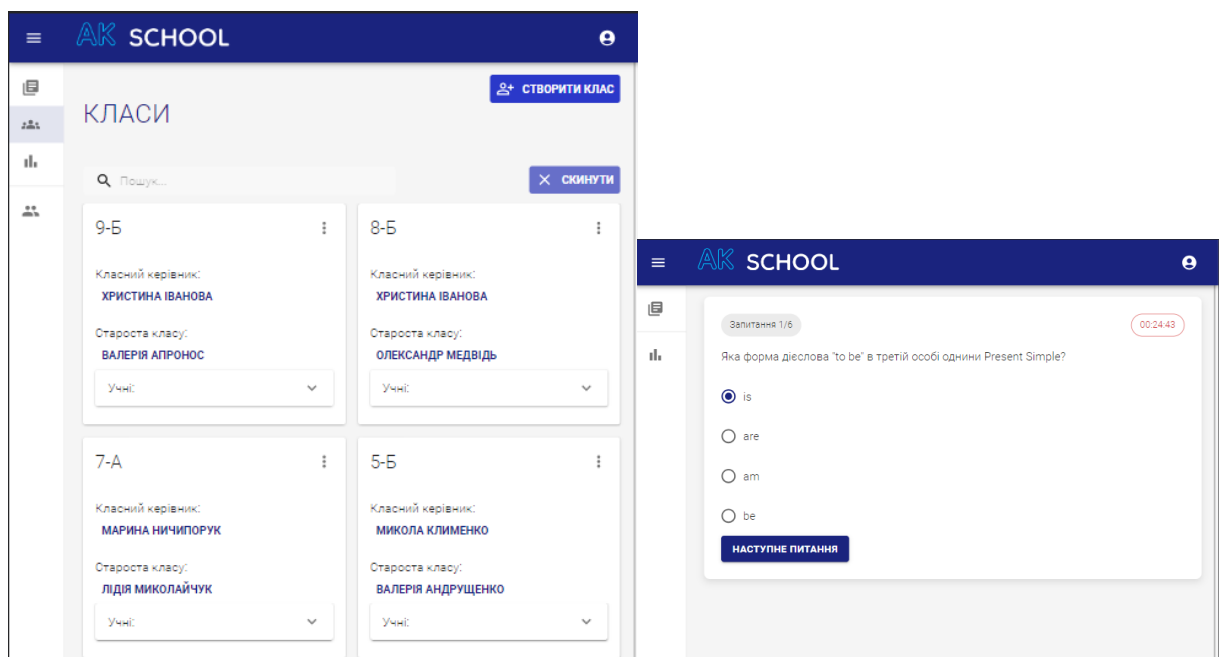


Рисунок 3.25 – Тест на планшеті

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	58
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Після цього ми перевіримо коректність відображення даних на телефоні

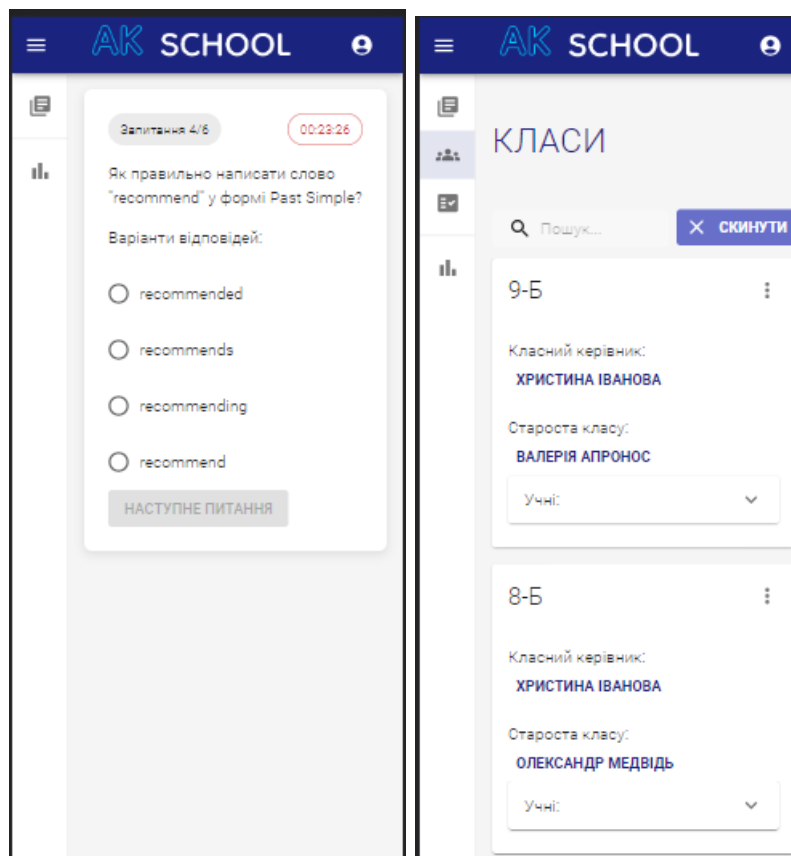


Рисунок 3.26 – Тест на телефоні

З вище представлених зображень, можна зробити висновок, що сайт успішно пройшов тестування адаптивності і здатний належним чином відображатися на різних розмірах екранів та пристроїв. Це дозволяє користувачам зручно використовувати сайт незалежно від пристрою, з якого вони отримують доступ, та забезпечує їм оптимальний досвід взаємодії.

Висновки до третього розділу

У даному розділі були встановлені мінімальні вимоги для безперебійного користування додатком. Це означає, що для використання додатку не потрібні спеціалізоване обладнання або додаткові налаштування, окрім стандартних компонентів, які зазвичай присутні на більшості сучасних операційних систем.

Крім того, у цьому розділі була надана схема розгортання розробленої системи, що демонструє, як різні компоненти взаємодіють між собою.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	59
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

Опис інтерфейсу розробленого програмного продукту також був включений до цього розділу. Було проведено детальний опис структури інтерфейсу додатку та послідовність його використання. Були описані основні вікна додатку, а також надано детальні описи алгоритмів їх функціонування. Це дозволило користувачам отримати чітке розуміння процесу роботи з кожним вікном та етапами, які потрібно пройти для досягнення бажаного результату.

Необхідні тестування додатку були успішно проведені, що дозволило підтвердити його коректну роботу та відповідність вимогам. Таким чином, система працює надійно та ефективно, здатна виконувати поставлені завдання згідно з встановленими вимогами.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		60

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано інформаційну систему з підтримки дистанційної форми навчання для загальноосвітньої школи (робота з користувачами, створення тестувань). Для цього було вирішено наступні завдання:

1. Було проведено аналіз напрямків використання інформаційних технологій для розробки системи для дистанційної форми навчання. Це дозволило визначити основні тенденції та можливості в цій галузі.

2. В ході аналізу аналогів програмного продукту були виявлені подібні системи для дистанційного навчання. Це дало змогу виявити переваги та недоліки існуючих рішень і врахувати їх при розробці нової системи.

3. Була обрана архітектура системи, яка задовольняє основні вимоги проекту, забезпечує ефективну організацію навчального процесу та зручний інтерфейс для користувачів.

4. На основі обраної архітектури та описаних вимог були обрані інструментальні засоби, що забезпечують ефективну розробку, зручну роботу з кодом, швидке завантаження даних та створення привабливого користувацького інтерфейсу.

5. В ході розробки самої інформаційної системи була розроблена структура бази даних, що дозволяє зберігати необхідну інформацію для коректної роботи системи. В результаті, був розроблений додаток, який відповідає поставленим вимогам, що підтверджує його надійну та ефективну роботу.

Усі ці кроки та етапи розробки системи для дистанційної форми навчання дозволили створити функціональний, зручний та надійний програмний продукт, який задовольняє потреби користувачів у дистанційному навчанні.

У подальшому програмний продукт може розширюватися шляхом введення нових функцій, поліпшення інтерфейсу, розробки мобільного додатку, аналітики та звітності, а також інтеграції з іншими системами. Це покращить його функціональність і зручність у користуванні.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система управління навчанням [Електронний ресурс] – Режим доступу до ресурсу: <https://www.human.ua/>.
2. Система для створення персоналізованого навчального середовища Moodle [Електронний ресурс] – Режим доступу до ресурсу: <https://moodle.org>.
3. Безкоштовний веб-сервіс для навчальних закладів [Електронний ресурс] – Режим доступу до ресурсу: <https://classroom.google.com/>.
4. Розділяй та володарюй: що таке патерни MVC і MVP, та як їх використовувати [Електронний ресурс] – Режим доступу до ресурсу: <https://highload.today/uk/blogs/shho-take-mvc-ta-mvp-patterni/>.
5. Що таке SPA-додатки [Електронний ресурс] – Режим доступу до ресурсу: <https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya>.
6. Офіційна документація HTTPS [Електронний ресурс] – Режим доступу до ресурсу: <http://surl.li/wxpq>.
7. Посібник із MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/sql/mysql/>.
8. Документація по використанню PHPMyAdmin [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.phpmyadmin.net/uk/latest/>.
9. А.Зандстра, Ф.Матт. PHP Objects, Patterns, and Practice, 2013. Режим доступу: <https://www.apress.com/la/book/9781430260325>.
10. Офіційна документація фреймворку Laravel [Електронний ресурс] – Режим доступу до ресурсу: <https://laravel.com/>.
11. Встановлення OpenServer [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://ospanel.io/docs/>.
12. Сайт JavaScript бібліотека для створення інтерфейсів користувача React [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.reactjs.org/>.
13. Сучасний підручник з JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.javascript.info/>.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	62
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

14. Material-UI - бібліотека компонентів [Електронний ресурс] – Режим доступу до ресурсу: <https://mui.com/material-ui/getting-started/overview/>.
15. Build a React App With a Laravel Back End [Електронний ресурс] – Режим доступу до ресурсу: <https://code.tutsplus.com/tutorials/build-a-react-app-with-laravel-backend-part-2-react--cms-29443>.
16. Побудова діаграми використання [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.
17. Побудова діаграми класів [Електронний ресурс] – Режим доступу до ресурсу: https://flexberry.github.io/ru/gpg_class-diagram.html.
18. StarUML documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.staruml.io/>.
19. Побудова діаграми діяльності [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>.
20. Вступ | Axios Docs [Електронний ресурс] – Режим доступу до ресурсу: <https://axios-http.com/uk/docs/intro>.
21. Пакет для керувати правами та ролями користувачів [Електронний ресурс] – Режим доступу до ресурсу: <https://spatie.be/docs/laravel-permission/v5/introduction>.
22. Офіційна документація Formik [Електронний ресурс] – Режим доступу до ресурсу: <https://formik.org/>.
23. Офіційна документація Yup [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/yup>.
24. Офіційна документація Redux Toolkit [Електронний ресурс] – Режим доступу до ресурсу: <https://redux-toolkit.js.org/introduction/getting-started>.
25. Побудова діаграми розгортання [Електронний ресурс] – Режим доступу до ресурсу: <https://creately.com/guides/deployment-diagram-tutorial/>.

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	63
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		64

1. AnswersController

```

class AnswersController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StoreAnswerRequest $request)
    {
        $validated = $request->validated();
        $studentAnswers = $validated['answers'];
        unset($validated['answers']);

        $answerCreated = Answer::create($validated);
        $totalScore = 0;

        foreach ($studentAnswers as $questionId => $studentAnswer) {
            $question = Questions::where(['id' => $questionId, 'test_id' =>
                $validated['test_id']])->first();
            if (!$question) {
                return response("Invalid question ID:\"$questionId\"", 400);
            }

            $scoreForQuestion = $question->checkAnswer($studentAnswer);
            $data = [
                'question_id' => $questionId,
                'answer_id' => $answerCreated->id,
                'answer' => is_array($studentAnswer) ? json_encode($studentAnswer) :
                $studentAnswer,
                'score' => $scoreForQuestion,
            ];

            $totalScore += $scoreForQuestion;
            QuestionAnswer::create($data);
        }

        $test = Test::find($validated['test_id']);
        $maximumScore = $test->maximumScoreForTest();

        $answerCreated->total_score = 12 * ($totalScore / $maximumScore);
        $answerCreated->save();
    }
}

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		65

```

        return response(new TestResource($test), 201);
    }
}

2. AuthController
class AuthController extends Controller
{

    public function login(LoginRequest $request)
    {
        $credentials = $request->validated();
        if (!Auth::attempt($credentials)) {
            return response([
                'message' => 'Вказаний логін або пароль невірні'
            ], 422);
        }
        /** @var User $user */
        $user = Auth::user();
        $token = $user->createToken('main')->plainTextToken;
        $user->changeOfStatus();
        return response([
            'user' => new UserResource($user),
            'token' => $token,
        ]);
    }

    public function logout(Request $request)
    {
        $user = Auth::user();
        $user->currentAccessToken()->delete();

        return response('', 204);
    }

    public function me(Request $request)
    {
        $user = $request->user();
        return response([
            'user' => new UserResource($user),
        ]);
    }
}

```

3. ClassesController

```

class ClassesController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {

        $user = auth()->user();
    }
}

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		66

```

// Перевірка ролі користувача
if ($user->hasRole('admin')) {
    // Якщо користувач є адміном, показуємо всі класи
    $classes = Classes::query()->orderBy('name', 'desc')->get();
} else if ($user->hasRole('teacher')) {
    // Якщо користувач є вчителем, показуємо тільки ті класи, в яких він є класним
керівником
    $teacherClasses = $user->teacherClasses()->orderBy('name', 'desc')->get();
    $classesSubject = $user->teacherSubjects()->with('classes')->get()-
>pluck('classes')->flatten();
    $classes = $teacherClasses->merge($classesSubject);
} else {
    // Якщо роль користувача не відповідає жодній з вище перерахованих, показуємо
порожній список класів
    $classes = [];
}

return ClassResource::collection($classes);
}

/**
 * Store a newly created resource in storage.
 */
public function store(StoreClassesRequest $request)
{
    $data = $request->validated();
    $students = $data['student_ids'];
    unset($data['student_ids']);

    $class = Classes::create($data);

    $students = collect($students)->map(function ($studentId) {
        return User::findOrFail($studentId);
    });

    $class->students()->saveMany($students);

    return response(new ClassResource($class), 201);
}

/**
 * Update the specified resource in storage.
 */
public function update(UpdateClassesRequest $request, Classes $class)
{
    $data = $request->validated();
    $students = $data['student_ids'];
    unset($data['student_ids']);

    $class->update($data);

    $students = User::whereIn('id', $students)->get();

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		67

```

        $class->students()->whereNotIn('id', $students->pluck('id'))->
>update(['class_id' => null]);

        $class->students()->saveMany($students);

        return response(new ClassResource($class), 201);
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Classes $class)
    {
        $class->delete();
        return response()->json(['message' => `Class  deleted successfully`], 200);
    }
}

```

4. TestController

```

class TestController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $user = auth()->user();

        // Перевірка ролі користувача
        if ($user->hasRole('admin')) {
            // Якщо користувач є адміном, показуємо всі класи
            $tests = Test::query()->orderBy('title', 'desc')->get();
        } else if ($user->hasRole('teacher')) {
            // Якщо користувач є вчителем, показуємо тільки ті класи, в яких він є класним
керівником
            $tests = $user->teacherTests()->orderBy('title', 'desc')->get();
        } else {
            // Якщо роль користувача не відповідає жодній з вище перерахованих, показуємо
порожній список класів
            $tests = [];
        }

        return TestResource::collection($tests);
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StoreTestRequest $request)
    {
        $data = $request->validated();
    }
}

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		68

```

    $subjects= $data['subject_ids'];
    unset($data['subject_ids']);

    $test = Test::create($data);

    $subjects = collect($subjects)->map(function ($subjectId) {
        return Subject::findOrFail($subjectId);
    });

    $test->subjects()->attach($subjects->pluck('id'));

    return response(new TestResource($test), 201);
}

/**
 * Display the specified resource.
 */
public function show(string $id)
{
    //
}

/**
 * Update the specified resource in storage.
 */
public function update(UpdateTestRequest $request, Test $test)
{
    $data = $request->validated();
    $subjects = $data['subject_ids'];
    unset($data['subject_ids']);

    $test->update($data);

    $test->subjects()->sync($subjects);

    return response(new TestResource($test), 201);
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(Test $test)
{
    $test->delete();

    return response('test was deleted', 204);
}

public function changeTestStatus(Test $test)
{
    if ($test->is_active === 0) {
        // Перевіряємо, чи тест має хочаб 2 питання

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		69

```

        if ($test->questions()->count() < 2) {
            return response()->json(['errors' => 'Тест повинен мати хочаб 2
питання.'], 400);
        }

        // Перевіряємо, чи дата завершення тесту ще не минула
        if ($test->end_time && Carbon::now()->isAfter($test->end_time)) {
            return response()->json(['errors' => 'Дата завершення тесту вже
минула.'], 400);
        }
    }

    // Виконуємо зміну статусу тесту
    $test->setStatus(!$test->is_active);

    return response()->json(['message' => 'Статус тесту змінено успішно.'], 200);
}

public function addQuestions(StoreQuestionsRequest $request, string $id)
{
    $data = $request->validated();
    if (is_array($data['options'])) {
        $data['options'] = json_encode($data['options']);
    }
    // var_dump($data);
    $question = Questions::create($data);

    $parsedQuestion = json_decode($question, true);
    $parsedQuestion['options'] = json_decode($parsedQuestion['options'], true);

    return response($parsedQuestion, 201);
}

public function updateQuestions(UpdateQuestionsRequest $request, string $id)
{
    $data = $request->validated();
    $question = Questions::findOrFail($id);
    if (is_array($data['options'])) {
        $data['options'] = json_encode($data['options']);
    }

    $question->update($data);

    $parsedQuestion = json_decode($question, true);
    $parsedQuestion['options'] = json_decode($parsedQuestion['options'], true);

    return response($parsedQuestion, 201);
}

public function deleteQuestions(string $id)
{
    $question = Questions::findOrFail($id);
    $question->delete();
}

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		70

```

        return response('question was deleted', 204);
    }
    public function getBySlug(Test $test)
    {
        //      if (!$test->is_active) {
        //          return response('', 404);
        //      }
        //      $currentDate = new \DateTime();
        //      $expireDate = new \DateTime($survey->expire_date);
        //      if ($currentDate > $expireDate) {
        //          return response('', 404);
        //      };
        return new TestResource($test);
    }
}

```

5. UserController

```

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        // $perPage = $request->input('per_page', 10);
        return UserResource::collection(
            User::query()->orderBy('id', 'desc')->get()
        );
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(StoreUserRequest $request)
    {
        $data = $request->validated();
        $data['password'] = bcrypt($data['password']);
        $data['status'] = 0;
        $role = $data['role'];
        unset($data['role']);

        $user = User::create($data);

        $role = Role::findByName($role, 'web'); // знайти роль за іменем
        $user->assignRole($role); // додати роль до користувача
        return response(new UserResource($user), 201);
    }

    /**
     * Display the specified resource.
     */
    public function show(User $user)
    {
    }
}

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		71

```

{
    //
}

/**
 * Update the specified resource in storage.
 */
public function update(UpdateUserRequest $request, User $user)
{
    $data = $request->validated();
    $role = $data['role'];
    unset($data['role']);

    $role = Role::findByName($role, 'web'); // знайти роль за іменем

    $user->syncRoles($role);

    $user->update($data);
    return new UserResource($user);
}

public function newPassword(Request $request, User $user)
{
    $newPassword = $request->input('password');

    if ($newPassword) {
        $newPassword = bcrypt($newPassword);
        $user->update(['password' => $newPassword]);
    }
    return response()->json(['message' => 'Пароль успішно оновлено'], 200);
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(User $user)
{
    $user->delete();

    return response('user was deleted', 204);
}

public function getStudentsForClass($classId = null)
{
    $studentsWithoutClass = User::where('class_id', null)
        ->whereHas('roles', function ($query) {
            $query->where('name', 'student');
        })
        ->get();

    if ($classId) {
        $class = Classes::with('students')->find($classId);
        if ($class) {

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		72

```

        $students = $class->students->merge($studentsWithoutClass);
    } else {
        $students = $studentsWithoutClass;
    }
} else {
    $students = $studentsWithoutClass;
}

return UserResource::collection($students);
}

}

```

6. Модель Answer

```

class Answer extends Model
{
    use HasFactory;
    protected $table = 'answers';
    public $timestamps = false;

    protected $guarded = [];

    public function test()
    {
        return $this->belongsTo(Test::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function questionAnswers()
    {
        return $this->hasMany(QuestionAnswer::class);
    }
}

```

7. Модель Classes

```

class Classes extends Model
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    protected $guarded = [];

    public function monitor()
    {
        return $this->belongsTo(User::class, 'monitor_id');
    }

    public function teacher()
    {
        return $this->belongsTo(User::class, 'teacher_id');
    }
}

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		73

```

    public function students()
    {
        return $this->hasMany(User::class, 'class_id');
    }
    public function subjects()
    {
        return $this->belongsToMany(Subject::class, 'subject_classes', 'class_id',
'subject_id');
    }
}

```

8. Модель QuestionAnswer

```

class QuestionAnswer extends Model
{
    use HasFactory;

    protected $table = 'question_answers';
    public $timestamps = false;
    protected $guarded = [];

    public function question()
    {
        return $this->belongsTo(Questions::class);
    }

    public function answer()
    {
        return $this->belongsTo(Answer::class);
    }
}

```

8. Модель Questions

```

class Questions extends Model
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    protected $table = 'questions';
    protected $guarded = [];

    public function checkAnswer($answer)
    {
        $options = json_decode($this->options, true);
        switch ($this->type) {
            case 'single-choice':
                $correctOption = collect($options)->firstWhere('isCorrect', true);
                $isAnswerCorrect = ($correctOption && $correctOption['text'] ===
$answer);

                if ($isAnswerCorrect) {
                    return $this->score;
                }
                break;
            case 'short-answer':

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		74

```

        $userAnswer = mb_strtolower(trim($answer)); // перетворення введеної
відповіді у нижній регістр та видалення зайвих пробілів
        $correctAnswers = array_map('mb_strtolower', array_map('trim',
$options));
        if (in_array($userAnswer, $correctAnswers)) {
            return $this->score;
        }
        break;
    case 'multiple-choice':
        $correctOptions = collect($options)->where('isCorrect', true)->all();
        $scoreForEachCorrectAnswer = $this->score / count($correctOptions);
        $totalScore = 0;
        $countCorrectAnswer = 0;
        foreach ($correctOptions as $value) {
            if (in_array($value['text'], $answer)) {
                $countCorrectAnswer++;
                $totalScore += $scoreForEachCorrectAnswer;
            }
        }
        if ($countCorrectAnswer === count($answer)) {
            return $totalScore;
        }
        break;
    case 'matching':
        $correctOptions = $options['correctAnswers'];
        $scoreForEachCorrectAnswer = $this->score / count($correctOptions);
        $totalScore = 0;
        $countCorrectAnswer = 0;

        foreach ($correctOptions as $value) {
            if (in_array($value, $answer)) {
                $countCorrectAnswer++;
                $totalScore += $scoreForEachCorrectAnswer;
            }
        }
        if ($countCorrectAnswer === count($answer)) {
            return $totalScore;
        }
        break;
    }
    return 0;
}

public function test()
{
    return $this->belongsTo(Test::class, 'test_id');
}
}

```

9. Модель Test

```

class Test extends Model
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles, HasSlug;
}

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	75
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

```

protected $table = 'tests';
protected $guarded = [];

public function getSlugOptions(): SlugOptions
{
    return SlugOptions::create()
        ->generateSlugsFrom('title')
        ->saveSlugsTo('slug');
}

public function teacher()
{
    return $this->belongsTo(User::class, 'created_by');
}

public function questions()
{
    return $this->hasMany(Questions::class, 'test_id');
}

public function setStatus(bool $isActive): void
{
    $this->is_active = $isActive;
    $this->save();
}

public function answers()
{
    return $this->hasMany(Answer::class);
}

public function maximumScoreForTest()
{
    return $this->questions()->sum('score');
}

public function subjects()
{
    return $this->belongsToMany(Subject::class, 'test_subjects');
}
}

```

10.Модель User

```

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    const STATUS_NEW = 0;
    const STATUS_ACTIVE = 1;

    public function changeOfStatus()

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		76

```

    {
        if ($this->status == 0) {
            $this->status = 1;
            $this->isDirty('status') && $this->save();
        }
    }

    public static function getStatusTitleAttribute($user)
    {
        $status = $user->status;

        if ($status == self::STATUS_NEW) {
            return [
                'title' => 'New',
                'backgroundColor' => '#aee6eb',
                'color' => '#00aec1'
            ];
        } elseif ($status == self::STATUS_ACTIVE) {
            return [
                'title' => 'Active',
                'backgroundColor' => '#9fd99f',
                'color' => '#55a71d'
            ];
        } else {
            return 'Unknown';
        }
    }

    protected $guarded = [];
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    // protected $fillable = [
    //     'login',
    //     'first_name',
    //     'second_name',
    //     'password',
    //     'status',
    //     'class_id'
    // ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

```

Л.В.Андрущенко

Т.А.Вакалюк

Зм. Арк. № докум. Підпис Дата

ІІЗ.КР.Б-121-23-ІІЗ

77

```

/**
 * The attributes that should be cast.
 *
 * @var array<string, string>
 */
protected $casts = [
    'email_verified_at' => 'datetime',
];

public function classes()
{
    return $this->belongsTo(Classes::class, 'class_id');
}

public function monitorClasses()
{
    return $this->hasMany(Classes::class, 'monitor_id');
}

public function teacherClasses()
{
    return $this->hasMany(Classes::class, 'teacher_id');
}

public function teacherTests()
{
    return $this->hasMany(Test::class, 'created_by');
}

public function teacherSubjects(){
    return $this->hasMany(Subject::class, 'teacher_id');
}
}

```

11.usersSlice

```

import {createAsyncThunk, createSlice} from "@reduxjs/toolkit";
import axiosClient from "../../axios-client.js";
import {loginUser} from "../currentUserSlice.js";
import {getComparator, stableSort} from "../../utils/filtering.js";
import {getRoles} from "../role/rolesSlice.js";

const initialState = {
    users: [],
    studentsWithoutClass: [],
    visibleData: [],
    meta: [],
    errors: null,
    isLoading: true,
    status: 'idle',
}

export const createUser = createAsyncThunk(

```

		Л.В.Андрущенко			ПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		78

```

"users/createUser",
async (payload, {rejectWithValue}) => {
  try {
    const res = await axiosClient.post('/users/create', payload);
    return res.data;
  } catch (error) {
    if (!error.response) {
      // Якщо немає відповіді від сервера
      return rejectWithValue("Не вдалося з'єднатися з сервером.");
    }
    const {errors} = error.response.data;
    return rejectWithValue(errors);
  }
}

);

export const updateUser = createAsyncThunk(
  "users/updateUser",
  async ({id, data}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.put(`/users/${id}`, data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const getUsers = createAsyncThunk(
  "users/getUsers",
  async (_, {rejectWithValue}) => {
    try {
      const res = await axiosClient.get(`/users`);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const fetchStudentsWithoutClass = createAsyncThunk(
  'users/fetchStudentsWithoutClass',

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		79

```

    async (classId , {rejectWithValue}) => {
      try {
        const res = await axiosClient.get(`/students-without-class/${classId}`);
        return res.data;
      } catch (error) {
        if (!error.response) {
          // Якщо немає відповіді від сервера
          return rejectWithValue("Не вдалося з'єднатися з сервером.");
        }
        const {errors} = error.response.data;
        return rejectWithValue(errors);
      }
    }
  );
export const deleteUser = createAsyncThunk(
  "users/deleteUser",
  async ({id, user}, {rejectWithValue}) => {
    try {
      await axiosClient.delete(`/users/${id}`, user);
      return user;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);
export const generateNewPassword = createAsyncThunk(
  "users/generateNewPassword",
  async ({id, user}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.post(`/users/new-password/${id}`, user);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

const usersSlice = createSlice({
  name: "users",
  initialState,
  reducers: {

```

Л.В.Андрущенко

Т.А.Вакалюк

		Л.В.Андрущенко		
		Т.А.Вакалюк		
Зм.	Арк.	№ докум.	Підпис	Дата

ІІЗ.КР.Б-121-23-ПЗ

```

clearErrors: (state) => {
  state.errors = null;
},
reset: (state) => {
  state.users = [];
  state.visibleData = state.users;
  state.errors = null;
},
searchUsers: (state, {payload}) => {
  state.visibleData = state.users.filter(
    (item) =>
      item.first_name.toLowerCase().includes(payload.toLowerCase()) ||
      item.second_name.toLowerCase().includes(payload.toLowerCase()) ||
      item.login.toLowerCase().includes(payload.toLowerCase())
  );
},
sortUsers: (state, {payload}) => {
  state.visibleData = stableSort(
    state.visibleData,
    getComparator(payload.order, payload.orderBy),
  );
},
},
extraReducers: (builder) => {
  builder.addCase(createUser.pending, (state) => {
    //state.isLoading = true;
  });
  builder.addCase(createUser.fulfilled, (state, action) => {
    state.status = 'succeeded';
    state.users.unshift(action.payload);
    state.visibleData.unshift(action.payload);
    //state.isLoading = false;
  });
  builder.addCase(createUser.rejected, (state, action) => {
    state.status = 'rejected';
    state.errors = action.payload;
    //state.isLoading = false;
  });
  builder.addCase(updateUser.pending, (state) => {
    //state.isLoading = true;
  });
  builder.addCase(updateUser.fulfilled, (state, action) => {
    state.status = 'succeeded';
    state.users = state.users.map(user => user.id === action.payload.data.id ?
action.payload.data : user)
    state.visibleData = state.visibleData.map(user => user.id ===
action.payload.data.id ? action.payload.data : user)
  });
  builder.addCase(updateUser.rejected, (state, action) => {
    state.status = 'rejected';
    state.errors = action.payload;
  });
  builder.addCase(deleteUser.pending, (state) => {
    //state.isLoading = true;

```

Л.В.Андрущенко

Т.А.Вакалюк

ІІЗ.КР.Б-121-23-ІІЗ

81

Зм. Арк. № докум. Підпис Дата

```

    });
    builder.addCase(deleteUser.fulfilled, (state, action) => {
      state.status = 'succeeded';
      state.users = state.users.filter(user => user.id !== action.payload.id);
      state.visibleData = state.users;
    });
    builder.addCase(deleteUser.rejected, (state, action) => {
      state.status = 'rejected';
      state.errors = action.payload;
    });
    builder.addCase(getUsers.pending, (state) => {
      state.isLoading = true;
    });
    builder.addCase(getUsers.fulfilled, (state, action) => {
      state.status = 'succeeded';
      state.users = action.payload.data;
      state.visibleData = action.payload.data;
      state.meta = action.payload.meta;
      state.isLoading = false;
    });
    builder.addCase(getUsers.rejected, (state, action) => {
      state.status = 'rejected';
      state.errors = action.payload;
      state.isLoading = false;
    });
    builder.addCase(fetchStudentsWithoutClass.fulfilled, (state, action) => {
      state.status = 'succeeded';
      state.studentsWithoutClass = action.payload.data;
    });
  },
});

```

```
export const {clearErrors, searchUsers, sortUsers, reset} = usersSlice.actions;
```

```
export default usersSlice.reducer;
```

12.testsSlice

```

import {createAsyncThunk, createSlice} from "@reduxjs/toolkit";
import axiosClient from "../../axios-client.js";
import {createUser, deleteUser, getUsers, updateUser} from "../user/usersSlice.js";
import {deleteClass} from "../class/classesSlice.js";

```

```

const initialState = {
  test: {},
  tests: [],
  visibleData: [],
  errors: null,
  isLoading: true,
  status: 'idle',
}

```

```

export const createTest = createAsyncThunk(
  "tests/createTest",
  async (payload, {rejectWithValue}) => {

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		82

```

    try {
      const res = await axiosClient.post('/tests/create', payload);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const updateTest = createAsyncThunk(
  "tests/updateTest",
  async ({id, formData}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.put(`/tests/${id}`, formData);
      console.log(res.data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const deleteTest = createAsyncThunk(
  "tests/deleteTest",
  async ({id, test}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.delete(`/tests/${id}`);
      console.log(res);
      return test;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const getTests = createAsyncThunk(

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		83

```

"tests/getTests",
async (_, {rejectWithValue}) => {
  try {
    const res = await axiosClient.get('/tests');
    return res.data;
  } catch (error) {
    if (!error.response) {
      // Якщо немає відповіді від сервера
      return rejectWithValue("Не вдалося з'єднатися з сервером.");
    }
    const {errors} = error.response.data;
    return rejectWithValue(errors);
  }
}

);

export const getTestBySlug = createAsyncThunk(
  "tests/getTestBySlug",
  async ({slug}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.get(`tests/get-by-slug/${slug}`);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const changeTestStatus = createAsyncThunk(
  "tests/changeTestStatus",
  async ({id}, {rejectWithValue}) => {
    try {
      const res = await axiosClient.post(`/tests/${id}/changeTestStatus`);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		84

```

export const addQuestion = createAsyncThunk(
  "tests/addQuestion",
  async ({test_id, values}, {rejectWithValue}) => {
    try {
      console.log(values);
      const res = await axiosClient.post(`/tests/add-questions/${test_id}`,
values);
      console.log(res.data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const updateQuestion = createAsyncThunk(
  "tests/updateQuestion",
  async ({question_id, values}, {rejectWithValue}) => {
    try {
      console.log(values);
      const res = await axiosClient.put(`/tests/update-questions/${question_id}`,
values);
      console.log(res.data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера
        return rejectWithValue("Не вдалося з'єднатися з сервером.");
      }
      const {errors} = error.response.data;
      return rejectWithValue(errors);
    }
  }
);

export const createAnswer = createAsyncThunk(
  "tests/createAnswer",
  async (payload, {rejectWithValue}) => {
    try {
      console.log(payload);
      const res = await axiosClient.post('/answers', payload);
      console.log(res.data);
      return res.data;
    } catch (error) {
      if (!error.response) {
        // Якщо немає відповіді від сервера

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		85

```

        return rejectWithValue("Не вдалося з'єднатися з сервером.");
    }
    const {errors} = error.response.data;
    return rejectWithValue(errors);
}

});

export const deleteQuestion = createAsyncThunk(
    "tests/deleteQuestion",
    async ({id, question}, {rejectWithValue}) => {
        try {
            const res = await axiosClient.delete(`/tests/delete-questions/${id}`);
            return question;
        } catch (error) {
            if (!error.response) {
                // Якщо немає відповіді від сервера
                return rejectWithValue("Не вдалося з'єднатися з сервером.");
            }
            const {errors} = error.response.data;
            return rejectWithValue(errors);
        }
    }
);

const testsSlice = createSlice({
    name: "tests",
    initialState,
    reducers: {
        clearErrors: (state) => {
            state.errors = null;
        },
        reset: (state) => {
            state.tests = [];
            state.visibleData = state.classes;
            state.errors = null;
        },
        searchTest: (state, {payload}) => {
            state.visibleData = state.tests.filter(
                (item) =>
                    item.title.toLowerCase().includes(payload.toLowerCase())
            );
        },
        getTestById: (state, {payload}) => {
            state.test = state.visibleData.find(test => test.id === parseInt(payload));
        },
    },
    extraReducers: (builder) => {

        // tests //
        builder.addCase(createTest.pending, (state) => {

```

		Л.В.Андрущенко			ІПЗ.КР.Б-121-23-ПЗ	86
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        // state.isLoading = true;
    });
    builder.addCase(createTest.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.tests.unshift(action.payload);
        state.visibleData.unshift(action.payload);
    });
    builder.addCase(createTest.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(updateTest.pending, (state) => {
        // state.isLoading = true;
    });
    builder.addCase(updateTest.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.tests = state.tests.map(test => test.id === action.payload.id ?
action.payload : test)
        state.test = action.payload;
    });
    builder.addCase(updateTest.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(deleteTest.pending, (state) => {
        //state.isLoading = true;
    });
    builder.addCase(deleteTest.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.tests = state.tests.filter(test => test.id !== action.payload.id);
        state.visibleData = state.tests;
    });
    builder.addCase(deleteTest.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(changeTestStatus.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.test = {...state.test, is_active: !state.test.is_active};
    });
    builder.addCase(changeTestStatus.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(getTests.pending, (state) => {
        state.isLoading = true;
    });
    builder.addCase(getTests.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.tests = action.payload.data;
        state.visibleData = state.tests;
        state.isLoading = false;
    });
    builder.addCase(getTests.rejected, (state, action) => {

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		87

```

        state.status = 'rejected';
        state.errors = action.payload;
        state.isLoading = false;
    });
    builder.addCase(getTestBySlug.pending, (state) => {
        state.isLoading = true;
    });
    builder.addCase(getTestBySlug.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.test = action.payload.data;
        state.isLoading = false;
    });
    builder.addCase(getTestBySlug.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
        state.isLoading = false;
    });

    // questions //
    builder.addCase(addQuestion.pending, (state) => {
        // state.isLoading = true;
    });
    builder.addCase(addQuestion.fulfilled, (state, action) => {
        state.status = 'succeeded';
        console.log(action.payload);
        state.test.questions.push(action.payload);
    });
    builder.addCase(addQuestion.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(updateQuestion.pending, (state) => {
        // state.isLoading = true;
    });
    builder.addCase(updateQuestion.fulfilled, (state, action) => {
        state.status = 'succeeded';
        console.log(action.payload);
        state.test.questions = state.test.questions.map(question => question.id ===
action.payload.id ? action.payload : question);
    });
    builder.addCase(updateQuestion.rejected, (state, action) => {
        state.status = 'rejected';
        state.errors = action.payload;
    });
    builder.addCase(deleteQuestion.pending, (state) => {
        //state.isLoading = true;
    });
    builder.addCase(deleteQuestion.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.test.questions = state.test.questions.filter(question => question.id
!== action.payload.id);
        // state.visibleData = state.tests;
    });

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ІІЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		88

```

    builder.addCase(deleteQuestion.rejected, (state, action) => {
      state.status = 'rejected';
      state.errors = action.payload;
    });

    // answers //
    builder.addCase(createAnswer.pending, (state) => {
      // state.isLoading = true;
    });
    builder.addCase(createAnswer.fulfilled, (state, action) => {
      state.status = 'succeeded';
      state.tests = state.tests.map(test => test.id === action.payload.id ?
action.payload : test)
      state.test = action.payload;
    });
    builder.addCase(createAnswer.rejected, (state, action) => {
      state.status = 'rejected';
      state.errors = action.payload;
    });
  },
});

export const {clearErrors, reset, searchTest, getTestById} = testsSlice.actions;
export default testsSlice.reducer;

```

		Л.В.Андрущенко			ІІЗ.КР.Б-121-23-ПЗ	
		Т.А.Вакалюк				
Зм.	Арк.	№ докум.	Підпис	Дата		89