ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут заочно-дистанційного навчання
Форма навчання заочна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
*(підпис)*

«___» _____ 2023 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на тему**
**«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТЕСТУВАННЯ СТУДЕНТІВ З ДИСЦИПЛІНИ "АЛГОРИТМИ І СТРУКТУРИ ДАНИХ"»**

**зі спеціальності 122 Комп'ютерні науки**
**освітня програма «Комп'ютерні науки»**
**ступеня магістра**

**Виконавець роботи** Кобченко Марина Володимирівна
_____ «____»_____ 2023 р.
*(підпис)*

**Науковий керівник** к.ф.-м.н., доц., Черненко Оксана Олексіївна
_____ «____»_____ 2023 р.
*(підпис)*

**ПОЛТАВА 2023 р.**

**ЗАТВЕРДЖУЮ**
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 202__ р.

**ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК**
**ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

**на тему** «Розробка програмного забезпечення тестування студентів з дисципліни “Алгоритми і структури даних”»
зі спеціальності 122 Комп’ютерні науки
освітня програма «Комп’ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Кобченко Марина Володимирівна
Затверджена наказом ректора № ___-Н від «_» _____ 2023 р.
Термін подання студентом роботи «___»_____ 202__ р.
Вихідні дані до кваліфікаційної роботи: публікації по темі роботи, статті та документації з теми розробки програмного забезпечення тестування, стандарти.

Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП

1. ПОСТАНОВКА ЗАДАЧІ

2. ОГЛЯД СЕРВІСІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

   2.1. Онлайн-платформа Google Forms.

   2.2. Онлайн-платформа Kahoot.

   2.3. Онлайн-платформа Quizizz.

   2.4. Онлайн-платформа Classtime.

3. ТЕОРЕТИЧНА ЧАСТИНА

   3.1. Опис проектних рішень та інструментів розробки.

   3.2. Опис обраного підходу до створення програмного забезпечення.

4. ПРАКТИЧНА ЧАСТИНА

   4.1. Опис та побудова діаграми роботи системи та розгортання.

   4.2. Інструкція з використання програмного забезпечення.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

Перелік графічного матеріалу: 2 аркуша блок-схем, 35 ілюстрації.

Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали, посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Постановка задачі | Черненко О.О. | | |
| Інформаційний огляд | Черненко О.О. | | |
| Теоретична частина | Черненко О.О. | | |
| Практична частина | Черненко О.О. | | |

Календарний графік виконання кваліфікаційної роботи

| Зміст роботи | Термін виконання | Фактичне виконання |
|---|---|---|
| 1. Вступ | | |
| 2. Вивчення методичних рекомендацій та стандартів та звіт керівнику | | |
| 3. Постановка задачі | | |
| 4. Інформаційний огляд джерел бібліотек та інтернету | | |
| 5. Теоретична частина | | |
| 6. Практична частина | | |
| 7. Закінчення оформлення | | |
| 8. Доповідь студента на кафедрі | | |
| 9. Доробка (за необхідністю), рецензування | | |

Дата видачі завдання «__» _____ 202___ р.

Здобувач вищої освіти _____ Кобченко Марина Володимирівна
(підпис)

Науковий керівник _____ к.ф.-м.н., доц. Черненко О.О.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

**Результати захисту кваліфікаційної роботи**

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2023 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

**Затверджую**
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

**Погоджено**
Науковий керівник _____
к.ф.-м.н. Оксана ЧЕРНЕНКО
«____» _____ 202__ р.

**План**

кваліфікаційної роботи на тему
«Розробка програмного забезпечення тестування студентів з дисципліни “Алгоритми і структури даних”»
зі спеціальності 122 Комп’ютерні науки
освітня програма 122 «Комп’ютерні науки»
ступеня магістр
Прізвище, ім’я, по батькові Кобченко Марина Володимирівна

ВСТУП
1. ПОСТАНОВКА ЗАДАЧІ
2. ОГЛЯД СЕРВІСІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ
    2.1. Онлайн-платформа Google Forms.
    2.2. Онлайн-платформа Kahoot.
    2.3. Онлайн-платформа Quizizz.
    2.4. Онлайн-платформа Classtime.
3. ТЕОРЕТИЧНА ЧАСТИНА
    3.1. Опис проектних рішень та інструментів розробки.
    3.2. Опис обраного підходу до створення програмного забезпечення.
4. ПРАКТИЧНА ЧАСТИНА
    4.1. Опис та побудова діаграми роботи системи та розгортання.
    4.2. Інструкція з використання програмного забезпечення.
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ Марина КОБЧЕНКО
«____» _____ 202__ р.

# РЕФЕРАТ

**Записка**: 64 сторінки, 35 рисунків, 1 додаток, 21 літературне джерело.

**Мета роботи** – розробка програмного забезпечення тестування студентів з дисципліни «Алгоритми і структури даних».

**Об’єкт розробки** – процес проходження тестів та оцінки знань за допомогою системи онлайн-тестування.

**Методи дослідження** – використання технологій та інструментів розробки програмного забезпечення: фреймворк Next.js, мова програмування TypeScript, CSS-препроцесор Sass, редактор коду Visual Studio Code, розподілена система контролю версій Git, база даних MongoDB та бібліотеки React, Zod.

Сформульовано вимоги до програмного забезпечення. Зроблено огляд систем аналогічного призначення, виділені їх основні переваги. Виконано опис проектних рішень та інструментів до розробки програмного забезпечення. Обрано методологію створення програмного забезпечення. Побудовано діаграму прецедентів та розгортання. Розроблено програмне забезпечення для тестування студентів. Розроблено інструкцію з користування програмного забезпечення.

# ЗМІСТ

ВСТУП.....7
1. ПОСТАНОВКА ЗАДАЧІ.....9
2. ОГЛЯД СЕРВІСІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ.....11
2.1. Онлайн-платформа Google Forms.....11
2.2. Онлайн-платформа Kahoot.....13
2.3. Онлайн-платформа Quizizz.....15
2.4. Онлайн-платформа Classtime.....17
3. ТЕОРЕТИЧНА ЧАСТИНА.....19
3.1. Опис проектних рішень та інструментів розробки.....19
3.2. Опис обраного підходу до створення програмного забезпечення.....30
4. ПРАКТИЧНА ЧАСТИНА.....32
4.1. Опис та побудова діаграми роботи системи та розгортання .....32
4.2. Інструкція з використання програмного забезпечення.....35
ВИСНОВКИ.....45
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....46
ДОДАТОК А. ВИХІДНІ КОДИ.....48

# ВСТУП

*Актуальність:* на сьогоднішній день питання дистанційного навчання є досить актуальним, адже воно дозволяє здобувати знання у будь-якому місці, де є можливість виходу в Інтернет. Дистанційне навчання не тільки надає можливість легко та у будь-який зручний час здобувати знання, а і оцінювати ці знання за допомогою різних інструментів, таких як онлайн-тестування.

Онлайн-тестування є досить важливою складовою такої системи, що дає змогу перевірити та закріпити знання з боку студента та оцінити їх з боку викладача. Цей інструмент є досить зручним, адже він дозволяє студенту проходити тести не будучи присутнім на занятті та у будь-який зручний час, що спрощує навчання і дає можливість підлаштувати його під особистий графік. Для викладача ця складова системи є також дуже зручною, бо вона дозволяє легко додавати та організовувати питання та відповіді в тесті, а також автоматизовано оцінювати відповіді студентів та формувати статистику за результатами проходження. Але основним призначенням онлайн-тестування є – перевірка власних знань студента, що допоможе зрозуміти рівень засвоєння знань швидко та автоматично.

Отже, в сучасному світі, тема розробки та використання сервісів онлайн-тестування є дуже актуальною.

Зважаючи на вказану вище актуальність, було вирішено розробити програмне забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних», а саме сервіс, що повинен складатися з таких частин: авторизація користувача, створення тестів, перегляд інформації по тестам, керування тестами та проходження цих тестів студентами.

*Метою роботи* є розробка програмного забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних», а саме сервіс, що повинен складатися з таких частин: авторизація, створення та керування тестами, перегляд інформації по тестам, проходження тестів.

*Об’єктом розробки* є процес проходження тестів та оцінки знань за допомогою сервісу онлайн-тестування.

*Предметом розробки* є програмне забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних», а саме сервіс з онлайн-тестування, що повинен складатися з таких частин: авторизація, створення та керування тестами, перегляд інформації по тестам, проходження тестів.

Кваліфікаційна робота складається з чотирьох розділів, а саме: постановка задачі, огляд сервісів аналогічного призначення, теоретична частина, практична частина.

Результатом виконання кваліфікаційної роботи є створення програмного забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних».

Обсяг пояснювальної записки: 64 сторінки, 35 рисунків, 1 додаток, 21 літературне джерело.

# 1. ПОСТАНОВКА ЗАДАЧІ

Потрібно розробити програмне забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних» у вигляді веб-додатку. Його веб-додатку має включати в себе розробку клієнтської частини, серверної частини та бази даних. Тобто, в результаті проведеної роботи, має бути окремий сайт, що буде виступати в ролі сервісу для проведення тестувань, який буде комунікувати з віддаленим сервером, який в свою чергу, буде оброблювати дані з використанням бази даних, що забезпечить належну роботу даного веб-додатку.

Реалізація розроблюваного проекту має передбачати створення наступних частин веб-додатку:

*Авторизація.* Авторизація передбачає можливість користувача реєстрації в системі через відповідну форму, яка повинна валідувати введенні дані і, після підтвердження, створювати обліковий запис та надавати доступ користувачу до системи. Окрім реєстрації, також повинна бути можливість входу користувача в систему, через відповідну форму, яка повинна також валідувати введенні дані користувача і надавати доступ до системи в разі вводу коректних даних. Валідація повинна проводитися як на стороні клієнта, так і на стороні серверу, щоб забезпечити коректність даних, що передаються. Після успішної авторизації користувача в системі, у нього повинна бути можливість вийти з власного облікового запису, що автоматично повинно перенаправити його на сторінку авторизації.

*Перегляд, створення та керування тестами.* Повинні бути реалізовані сторінки для перегляду всіх власних створених, активних та заархівованих тестів. Сторінка з всіма створеними тестами має передбачати можливість додавання нового тесту та перегляду інформації про уже додані. Для додавання нового тесту повинна бути відповідна кнопка, після натискання якої буде здійснено перехід до редактора тесту, де потрібно передбачати: введення назви тесту та створення набору питань і відповідей. Для перегляду інформації

повинна бути можливість натискання на потрібний тест, в результаті чого користувача буде переводити на сторінку, де можна переглянути питання та відповіді тесту, розпочати тестування, редагувати тест або видалити його. Сторінка з активними тестами має передбачати можливість перегляду інформації про розпочаті тести, шляхом натискання на потрібний і переходу на інформаційну сторінку, де повинна бути інформація про кожного учасника тестування та його оцінку. Сторінка із заархівованими тестами повинна містити всі завершені тести та інформацію про результати всіх учасників, яку можна переглянути після натискання на потрібний тест.

*Проходження тесту.* Якщо користувач розпочав тестування, то на екрані повинне з’явитися модальне вікно, що має містити QR-код та посилання на тест. Після переходу за посиланням, користувач повинен ввести власне ім’я та прізвище, після чого його має перенаправити до сторінки проходження тесту. В кінці проходження тесту користувача має перенаправити на сторінку з результатом, де він повинен мати змогу переглянути інформацію про правильність власних відповідей та отриману оцінку.

# 2. ОГЛЯД СЕРВІСІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

На даний момент у світі існує значна кількість безкоштовних платформ для проведення тестування, а далі ми розглянемо про деякі з них.

## 2.1. Онлайн-платформа Google Forms

Google Forms − це безкоштовна платформа, яка надає можливість створювати опитування, форми, реєстрації для івентів, тести та отримання зворотного зв'язку. Вся інформація, яка вноситься респондентами, автоматично потрапляє до Google Таблиць і завдяки такій функції можна швидко проаналізувати дані з мінімальними витратами часу і зусиль (Рисунок 2.1).

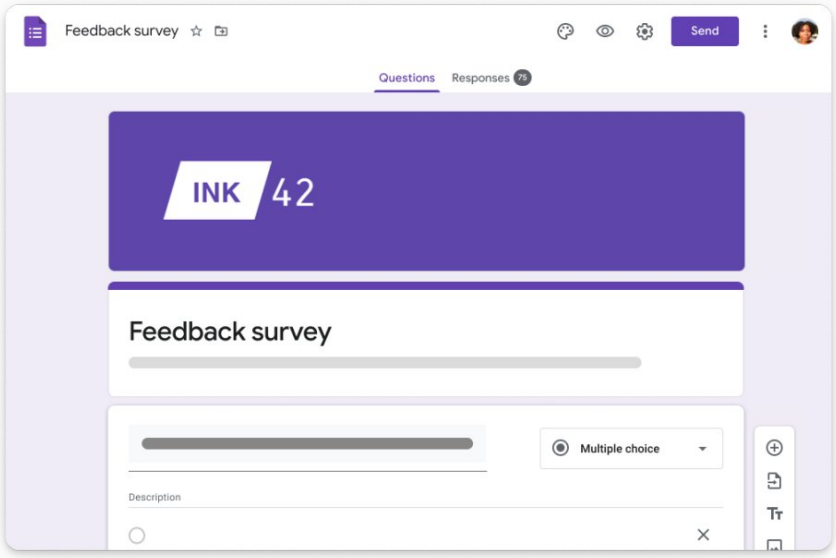


Рисунок 2.1 – Веб-інструмент «Google Forms»

У Google Forms є можливість створювати свої власні форми з різними типами питань, такими як: питання з одним варіантом відповіді, питання з кількома варіантами відповіді, відкриті питання, запити на завантаження файлів та багато інших (Рисунок 2.2).

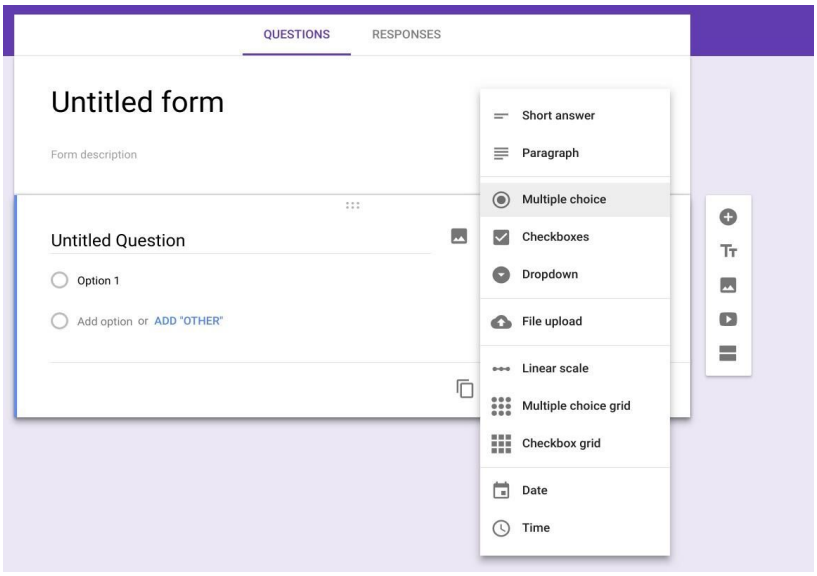


Рисунок 2.2 – Створення питань в «Google Forms»

Після того, як форма створена, є можливість надіслати посилання на неї іншим користувачам, які можуть відповісти на питання у формі.

Додаток має простий і лаконічний дизайн і для користувачів доступні різні приклади і шаблони в Google Forms, на основі яких ви можете легко створювати власні варіанти анкет, форм для реєстрації на заходи.

До переваг можна віднести:

- створення різних типів питань;
- налаштування параметрів форми;
- додавання логіки до форми;
- надсилання форми та збір відповідей;
- аналіз відповідей;
- використання готових шаблонів;
- інтеграція з іншими продуктами Google;
- безкоштовність.

Отже, Google Forms є потужним та зручним інструментом для створення опитувань, анкет, тестів та інших типів форм, який може бути використаний в різних контекстах, від бізнесу та освіти до особистих потреб [1].

## 2.2. Онлайн-платформа Kahoot

Kahoot − це онлайн-платформа, яка стала популярним інструментом для навчання та співпраці в класі або на відстані. Заснований на принципі гри, Kahoot надає вчителям можливість створювати інтерактивні квізи, ігри та опитування, які залучають учнів активному навчанню (Рисунок 2.3).


Рисунок 2.3 – Онлайн-платформа «Kahoot»

Сервіс підходить для:

- групового та дистанційного навчання викладачів та учнів;
- створення інтерактивних презентацій для роботи;
- домашнього навчання під час сімейного відпочинку.

Одна з ключових особливостей Kahoot − це його інтерактивний та забавний формат гри, який залучає учнів до активної участі в навчальному процесі. Учасники можуть відповідати на питання в режимі реального часу на своїх пристроях, таких як смартфони, планшети або комп'ютери, та бачити результати на екрані в реальному часі. Це створює емоційну та захоплюючу атмосферу, яка стимулює учнів до більш активної участі та залучення до навчання (Рисунок 2.4).

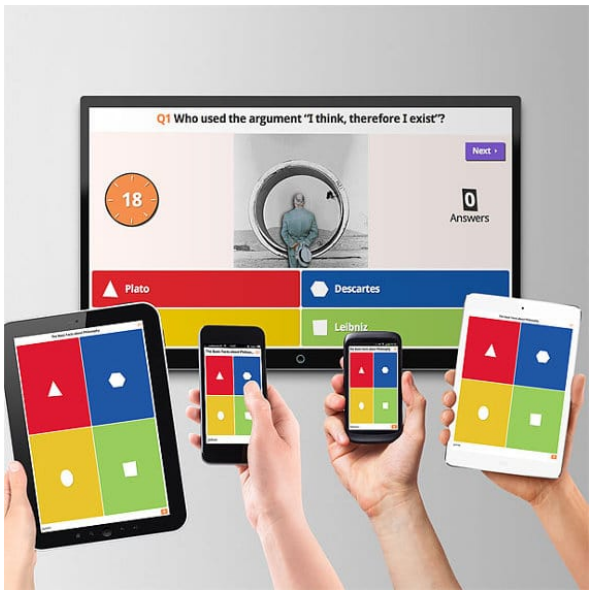


Рисунок 2.4 – Проходження тестування

Крім того, Kahoot є дуже гнучким інструментом, який дозволяє вчителям створювати власні тести, громадські опитування та вікторини відповідно до власних потреб та вимог навчального плану. Вчителі можуть налаштовувати питання, відповіді, додавати зображення та відео, тим самим створюючи цікавий та змістовний навчальний матеріал.

До переваг можна віднести:

- зрозумілий інтерфейс;
- створення різних типів тестів;
- створення тестів у форматі ігрових механік;
- ігрові механіки у кожному варіанті тесту чи вікторини;
- різноманітність тестів;
- режим реального часу;
- можливості швидкого створення тестів, опитувань, дискусій;
- можливості безкоштовної версії сервісу;
- бібліотека зображень, якою можна скористатися при створенні тестів;
- редактор математичних символів;
- зручна система звітів.

Загалом, Kahoot є потужним та відмінним інструментом для підвищення зацікавленості, залученості та результативності навчання. Він надає вчителям можливість створювати цікаві, інтерактивні та змагальні навчальні сесії, а учням − можливість активно брати участь в навчальному процесі та розвивати різні навички, такі як швидкість реакції, критичне мислення, мотивація та співпраця [2].

### 2.3. Онлайн-платформа Quizizz

Quizizz − це онлайн-платформа для створення інтерактивних квізів, тестів та інших освітніх активностей. Вона дозволяє вчителям, тренерам, студентам та іншим користувачам створювати власні тестові завдання, а також використовувати готові квізи, створені іншими користувачами (Рисунок 2.5).


Рисунок 2.5 – Онлайн платформа «Quizizz»

Quizizz дозволяє проводити онлайн-тести, ділитися посиланнями з учасниками, а також відстежувати прогрес та результати в режимі реального часу. Користувачі можуть використовувати Quizizz як для формальної, так і для неформальної освіти, включаючи шкільну освіту, вищу освіту, корпоративне навчання та інші види навчання. Quizizz також може використовуватися як засіб оцінювання знань, тренування пам'яті, розвитку навичок роботи з питаннями та відповідями, а також як забавний спосіб вивчення нового матеріалу.

Учні можуть надавати відповідь на тестові завдання використовуючи планшет, ноутбук, смартфон, тобто будь-який пристрій, що має доступ до Інтернету. Вчитель при створені тесту має можливість додати в запитання фотографії, відеофрагменти, формули. Швидкість виконання завдань вікторин, тестів керується автоматично системою, так як вчитель задає часові межі для кожного питання при їх створенні. Також вчитель вводить бали за правильні відповіді на запитання і додаткові бали за швидкість відповіді нараховує система. Під час тесту вчитель бачить процес проходження тесту кожним учнем на своєму пристрої (Рисунок 2.6).

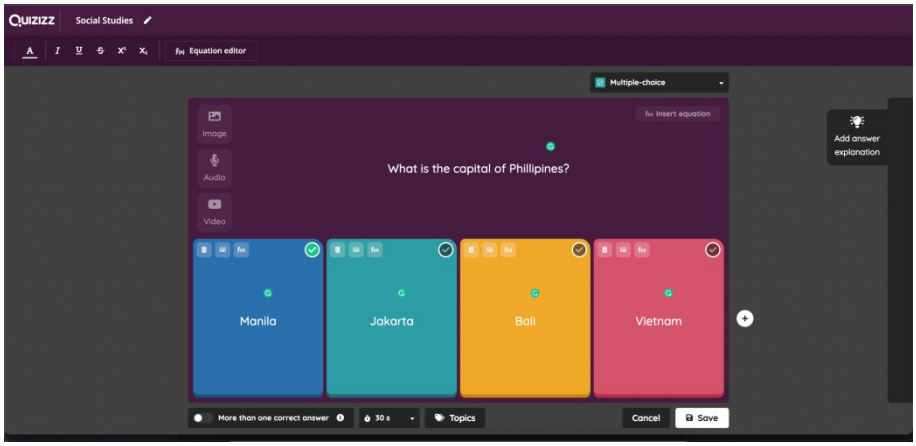


Рисунок 2.6 – Створення тесту

Основні особливості онлайн-платформи:

– Можливість створювати питання, які містять різноманітні типи відповідей, такі як багатовибірні питання, питання з вибором послідовності, відповіді на відкриті питання, та інші. Користувачі можуть вибирати з різних варіантів відповідей та отримувати миттєвий зворотний зв'язок на основі правильної відповіді.

– Можливість встановлювати таймер на кожне питання, обмеження на кількість спроб, додавати підказки та коментарі до питань, а також налаштовувати подробиці доступу до тесту.

– Кольоровий, привабливий та гармонічний дизайн, який робить процес вивчення цікавий та захоплюючим для учнів.

– Можливість відстежувати прогрес учнів, аналізувати результати тестів та отримувати звіти зі статистикою.

– Підтримка кількох мов, що дозволяє користувачам створювати та використовувати тести на різних мовах відповідно до своїх потреб.

– Співпраця та змагання, де учні можуть грати в командах, взаємодіяти між собою та конкурувати в режимі реального часу, що робить навчання веселим та захоплюючим [3].

## 2.4. Онлайн-платформа Classtime

Classtime – це українська онлайн-платформа для оцінювання знань та прогресу учнів дистанційно. Допомагає вчителям створювати та проводити інтерактивні уроки та тести в режимі реального часу (Рисунок 2.7).

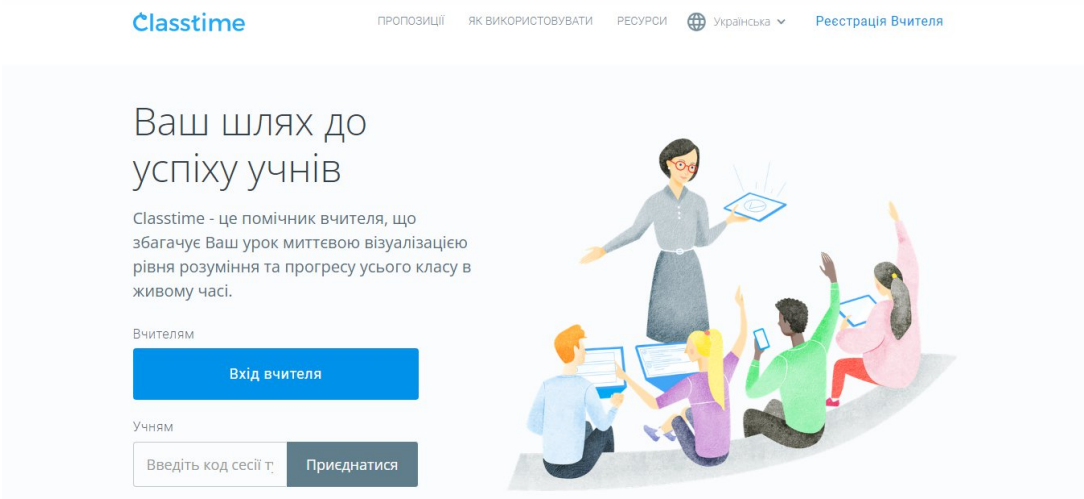


Рисунок 2.7 – Онлайн-платформа для оцінювання знань «Classtime»

Основні функції Classtime включають:

– створення власних завдання у зручному конструкторі, чи використання готових за шкільними предметами;

– автоматично оцінювати учнівські роботи й економити час і нерви на тестуваннях;

– проводити контрольні та самостійні роботи з налаштуваннями антисписування;

− додати гейміфікації й інтерактивності у вивчення нових тем і домашні роботи;

− опція, за допомогою якої викладач може перемішати запитання;

− організовувати роботу з учнями в поєднанні з Google Classroom.

До завдання можна додавати детальний опис, зображення, відео та аудіоматеріали, щоб учні могли зорієнтуватися в темі самостійно і вже потім проходити оцінювання. Сесії на Classtime відкриваються з будь-якого смартфона навіть з мобільним інтернетом, щоб учні завжди могли з ними впоратися.

Сервіс передбачає також платний функціонал. Додаткові функції доступні в платній версії сервісу, але за умови запрошення викладачів до роботи у даному сервісі, є можливість використовувати вказані функції безкоштовно.

Загалом, Classtime − це потужний інструмент для проведення тестів та оцінювання знань, який допомагає вчителям ефективно проводити оцінювання знань учнів, відстежувати їх прогрес та забезпечувати індивідуальний підхід до навчання [4].

# 3. ТЕОРЕТИЧНА ЧАСТИНА

## 3.1. Опис проектних рішень та інструментів розробки

За результатами аналізу систем аналогічного призначення, було обрано:

- мову програмування TypeScript;
- фреймворк Next.js;
- Css-препроцесор Sass;
- бібліотеки: React, Zod;
- редактор коду Visual Studio Code;
- система контролю версіями Git;
- база даних MongoDB.

TypeScript – це статично типізована, об'єктно-орієнтована, компільована мова програмування, яка базується на JavaScript. TypeScript дозволяє розробникам створювати більш структурований та безпечний код (Рисунок 3.1). Ця мова з'явилася у 2012 році, відразу ж після того, як компанія Microsoft опублікувала першу версію TypeScript. Розробником мови є Андерс Гейлсберг.

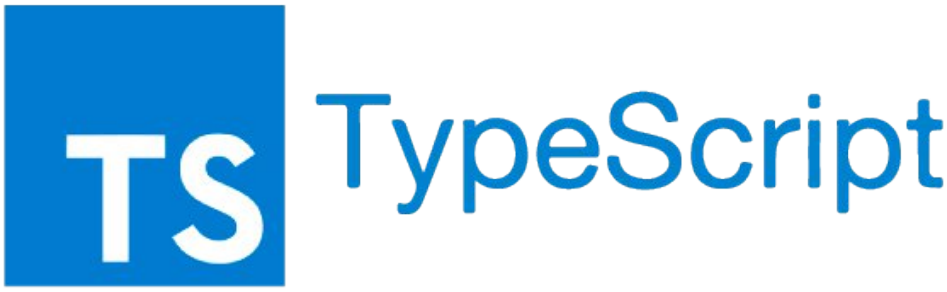


Рисунок 3.1 – Логотип мови програмування «TypeScript»

TypeScript дозволяє програмістам використовувати статичну типізацію, що означає, що кожна змінна, функція, параметр та інший елемент коду повинен мати певний тип даних. Наприклад, змінна може бути числом, рядком або булевим значенням. Це дозволяє виявляти помилки на етапі компіляції, перед тим, як програма запуститься, тим самим забезпечуючи більш високу якість коду та зменшуючи кількість помилок під час виконання програми.

Крім статичної типізації, TypeScript має багато інших функцій, які роблять його потужним інструментом для розробки програмного забезпечення. TypeScript підтримує класи, інтерфейси, модулі, збірку модулів та інші функції.

TypeScript використовується для розробки веб-додатків та інших програмних продуктів. Він добре підходить для великих проектів, де потрібно забезпечити якість та безпеку коду. TypeScript дозволяє зменшити кількість помилок та зробити розробку програмного забезпечення більш продуктивною та ефективнішим. Він також підтримується багатьма редакторами коду та інтегрованими середовищами розробки, що робить його дуже зручним для використання.

Переваги TypeScript:

- вивід помилок під час компіляції;
- багата документація;
- підтримка статичної типізації;
- підтримка ООП;
- підтримка стандартів ES6;
- має відкритий вихідний код [5].

Next.js – це фреймворк для створення веб-додатків з використанням React.js. Next.js надає багато корисних можливостей для побудови високопродуктивних, масштабованих та SEO-оптимізованих додатків (Рисунок 3.2).


Рисунок 3.2 – Фреймворк «Next.js»

Однією з головних переваг Next.js є те, що він рендере сторінки на стороні сервера. Це означає, що сервер може генерувати HTML-код для сторінки і надсилати його клієнту, а не клієнт генерує його за допомогою JavaScript. Це може покращити продуктивність та SEO додатку.

Next.js також включає в себе ряд інших функцій, які можуть бути корисними при створенні та розгортанні веб-додатків. Наприклад, він має автоматичне розподілення коду, що означає, що ваш додаток завантажуватиме лише код, необхідний для поточної сторінки, а не весь код одразу. Він також має вбудований сервер розробки та набір інструментів для розгортання додатку [6].

Основні переваги Next.js:

− Серверний рендеринг, який дозволяє отримати високу швидкість завантаження сторінок та поліпшити SEO, а також забезпечує кращу інтерактивність на стороні клієнта.

− Статична генерація контенту дозволяє зменшити навантаження на сервер та забезпечити швидке завантаження сторінок.

− Підтримка SSR та SSG дозволяє використовувати різні підходи для рендерингу сторінок, залежно від їх характеристик.

− Підтримка TypeScript.

− Підтримка CSS-модулів.

− Велика кількість плагінів та пакетів.

− Можливість легко отримувати дані з зовнішніх джерел, таких як бази даних, API-інтерфейси та файли JSON, що дозволяє розробникам зосередитися на розробці функціональності, а не на роботі з даними.

− Зручні засоби для маршрутизації та роутингу, що дозволяє легко створювати багатосторінкові веб-додатки з плавним переходом між сторінками.

Отже, Next.js є потужним та зручним фреймворком для розробки веб-додатків на React, який забезпечує швидку та легку розробку, оптимізацію сторінок та широкі можливості для розширення функціональності.

Sass − це мова препроцесорів CSS, яка дозволяє програмувати стилі для веб-сайту з більшою ефективністю та простотою (Рисунок 3.3). Sass був розроблений в 2006 році групою розробників, очолюваною Гаммом Брюном (Hampton Catlin). Sass став відповіддю на проблеми, які виникають при написанні CSS, зокрема на низьку ефективність та складність підтримки коду.


Рисунок 3.3 – Логотип «Sass»

Основні переваги Sass:

− Код Sass більш організований у порівнянні з CSS. Sass може виконувати ту саму роботу, використовуючи менше коду. Це робить код Sass набагато легшим для читання і розуміння, особливо у великих веб-проектах, в яких беруть участь кілька розробників.

− Код Sass можна використовувати багаторазово. Sass дозволяє використовувати змінні та фрагменти коду, які можна повторно використовувати знову і знову. Це економить розробникам багато часу і знижує ризик помилок у коді.

− Sass стабільний з моменту свого випуску у 2006 році. Sass підтримується як основними розробниками, так і великими технологічними компаніями [7].

Основні особливості Sass:

− підтримка змінних;
− вкладеність;
− міксіни;
− функції;

− використання математичних операцій.

Загалом, Sass дозволяє забезпечити більш організований та ефективний процес розробки стилів для веб-сайтів, дозволяючи зменшити кількість коду та зробити стилі більш читабельними та легко зрозумілими. Завдяки Sass, веб-розробники можуть швидко та легко змінювати стилі, змінювати кольори та розміри, додавати нові стилі та розширювати існуючі. Крім того, Sass дозволяє підключати стандартні бібліотеки стилів, такі як Bootstrap, що дозволяє зменшити час на написання власного коду та зосередитися на більш важливих завданнях розробки веб-сайту.

React − це бібліотека JavaScript з відкритим вихідним кодом, розроблена компанією Facebook. Він використовується для швидкого та ефективного створення інтерактивних користувацьких інтерфейсів і веб-додатків зі значно меншим обсягом коду, ніж при використанні звичайного JavaScript (Рисунок 3.4).


Рисунок 3.4 – Логотип JavaScript-бібліотеки «React»

У React є можливість розробляєте свої додатки, створюючи багаторазові компоненти, які можна уявити як незалежні блоки Lego. Ці компоненти є окремими частинами кінцевого інтерфейсу, які, будучи зібраними, формують весь користувацький інтерфейс додатку.

React використовує підхід створення так званих односторінкові додатки (SPA). Односторінковий додаток завантажує лише один HTML-документ при

першому запиті. Потім він оновлює певну частину, вміст або тіло веб-сторінки, яка потребує оновлення, використовуючи JavaScript.

У React замість того, щоб прямо маніпулювати реальним DOM, використовується віртуальний DOM. Virtual DOM (віртуальний DOM) − це концепція, яка лежить в основі React, яка допомагає збільшити продуктивність веб-додатків і спростити роботу з ними. У React замість того, щоб прямо маніпулювати реальним DOM, використовується віртуальний DOM. Кожен раз, коли змінюється стан React-компонентів, віртуальний DOM створює нове дерево. Далі, він порівнює це нове дерево з попереднім деревом і знаходить різницю між ними (Рисунок 3.5).

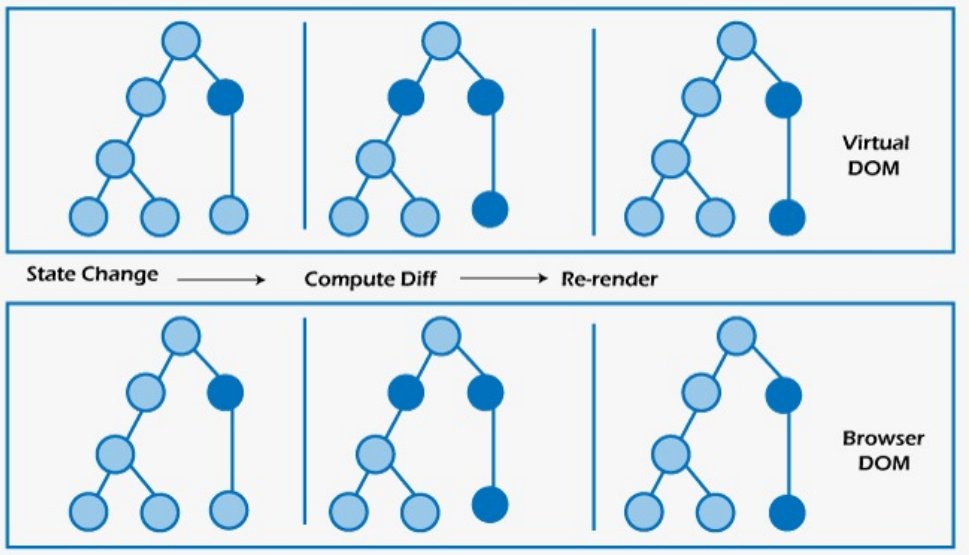


Рисунок 3.5 – Віртуальний DOM

На відміну від інших бібліотек, таких як Angular, React не запроваджує суворих правил щодо коду або організації файлів. Це означає, що розробники можуть вільно встановлювати свої правила, які їм найбільше підходять, і впроваджувати React так, як вони вважають за потрібне [8].

Основні переваги React:

− компонентна архітектура;
− віртуальний DOM;
− методи життєвого циклу;
− спільнота та документація.

Zod − це бібліотека оголошення та перевірки схем для TypeScript. Вона дозволяє створювати схеми для будь-яких даних, включаючи примітивні типи даних, об'єкти, масиви тощо. (Рисунок 3.6).


Рисунок 3.6 – Логотип TypeScript-бібліотеки «Zod»

Zod забезпечує валідацію схем для TypeScript під час виконання. Валідація схеми підтверджує, що дані відповідають стандарту, визначеному розробником. Це корисно при отриманні користувацьких даних у формах, надсиланні даних до API тощо. Зазвичай TypeScript забезпечує валідацію схеми лише під час компіляції, тоді як Zod забезпечує валідацію під час виконання [9].

Основні переваги використання Zod:

− Сильна типова перевірка. Виявляння багів на етапі компіляції і запобігає помилкам під час виконання програми.

− Модульність. Створення окремих схем для різних частин проекту і комбінування для складніших валідаційних сценаріїв.

− Підтримка асинхронної валідації. Можливість валідувати дані асинхронно, що дуже корисно для взаємодії зі зовнішніми джерелами даних.

− Підтримка різних типів даних. Об'єктів, масивів, чисел, рядків, булевих значення і багато інших.

Visual Studio Code − це безкоштовний та потужний текстовий редактор з відкритим вихідним кодом, що створений Microsoft, який доступний для Windows, macOS, Linux. Він має вбудовану підтримку JavaScript, TypeScript та

Node.js, а також багату екосистему розширень для інших мов програмування (C++, C#, Java, Python, PHP та Go), середовищ виконання (.NET та Unity) (Рисунок 3.7) [10].

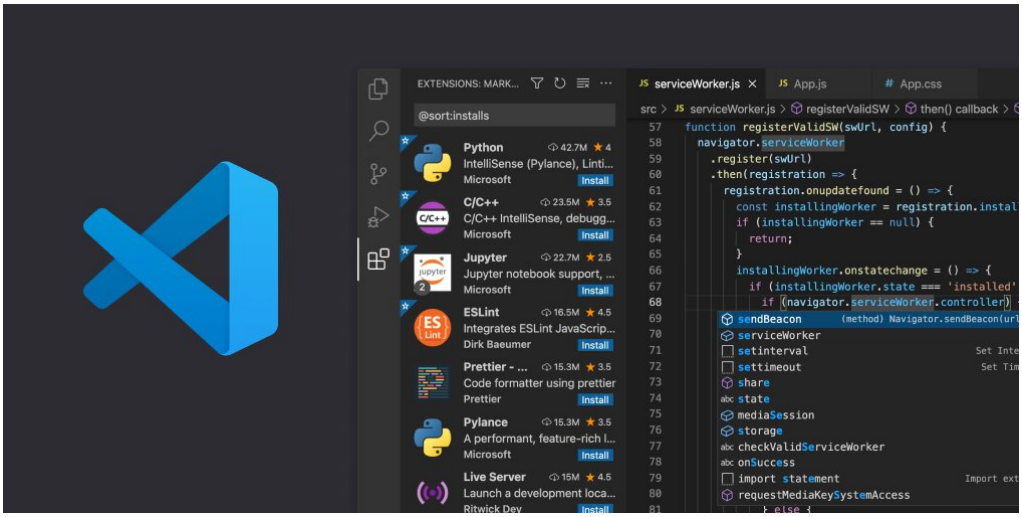


Рисунок 3.7 – Редактор коду «Visual Studio Code»

Visual Studio Code має кілька переваг для розробників:

– Кросплатформенність. Visual Studio Code доступний для Windows, macOS та Linux, що дозволяє розробникам використовувати один інструмент на будь-якій операційній системі.

– Висока продуктивність. Visual Studio Code має швидкий та потужний вбудований відладчик, що дозволяє розробникам ефективно відлагоджувати програмний код.

– Багатофункціональність. Visual Studio Code має багатофункціональний набір інструментів, таких як розширення, системи керування версіями та інші, що дозволяють розробникам працювати з будь-якими мовами програмування та інструментами розробки.

– Відкритий вихідний код. Visual Studio Code має відкритий вихідний код та має активну спільноту, що надає користувачам можливість зробити внесок у розвиток інструменту та допомогти вирішити проблеми.

– Інтеграція з іншими інструментами. Visual Studio Code інтегрується з різними інструментами, такими як GitHub, Docker та інші, що дозволяє розробникам працювати з різними сервісами та інструментами розробки.

Git − найпоширеніша система контролю версій. Він використовується для збереження та управління історією розробки програмного забезпечення, що дозволяє розробникам зберігати копії свого коду на різних комп'ютерах та спільно працювати над проектом, зберігаючи кожен крок розвитку проекту в історії змін. (Рисунок 3.8).

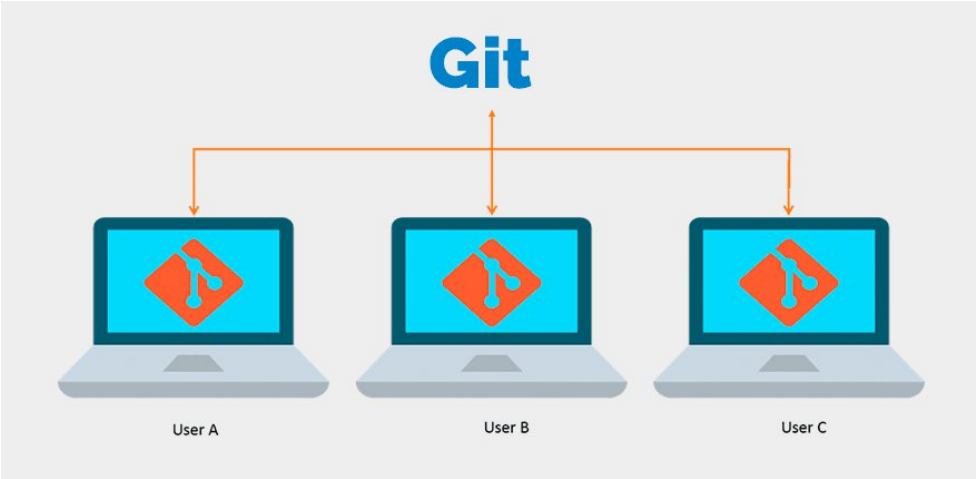


Рисунок 3.8 – Системи керування версіями файлів «git»

Git − це програмне забезпечення, яке працює локально. Файли та їхня історія зберігаються на комп’ютері. Є також можливість використовувати онлайн-хостинги (GitHub або Bitbucket) для зберігання копій файлів. Наявність централізованого місця, куди можна завантажувати свої зміни та завантажувати зміни інших, дозволяє легше співпрацювати з іншими розробниками. Git може автоматично об'єднувати зміни, тому двоє людей можуть навіть працювати над різними частинами одного файлу, а потім об'єднати ці зміни, не втрачаючи роботу один одного (Рисунок 3.9) [11].

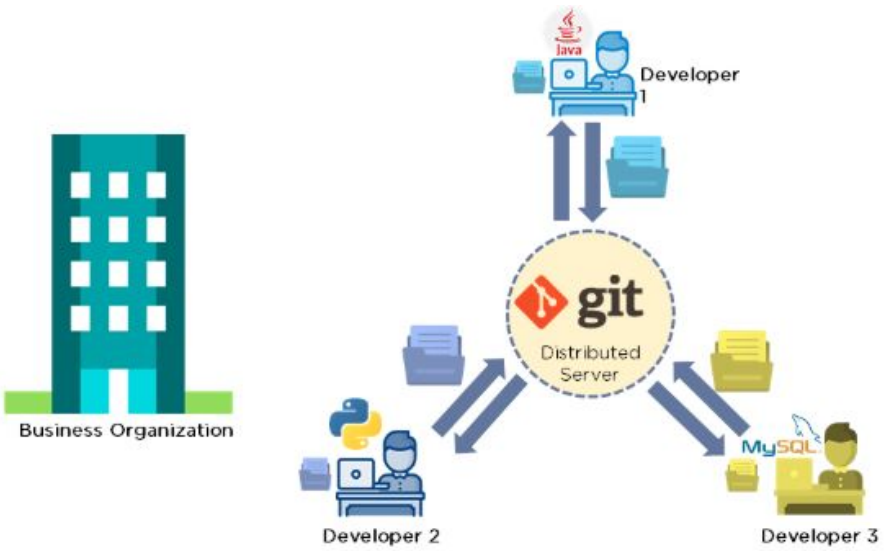


Рисунок 3.9 – Принцип роботи «git»

Основні переваги використання Git:

- керування версіями;
- робота з гілками;
- колективна робота;
- безпека файлів;
- кросплатформенність.

MongoDB − це популярна система керування базами даних (СКБД), яка використовує концепцію NoSQL. MongoDB використовується для зберігання великих обсягів даних, допомагаючи організаціям зберігати великі обсяги даних і при цьому швидко працювати. Організації також використовують MongoDB для спеціальних запитів, індексування, балансування навантаження, агрегації, виконання JavaScript на стороні сервера та інших функцій (Рисунок 3.10) [12].

Рисунок 3.10 – База даних «MongoDB»

Замість того, щоб використовувати таблиці та рядки, як у реляційних базах даних, як база даних NoSQL, архітектура MongoDB складається з колекцій та документів. Документи складаються з пар ключ-значення - основної одиниці даних MongoDB. Колекції, еквівалент таблиць SQL, містять набори документів. MongoDB пропонує підтримку багатьох мов програмування, таких як C, C++, C#, Go, Java, Python, Ruby та Swift.

Середовище MongoDB надає користувачам сервер для створення баз даних за допомогою MongoDB. MongoDB зберігає дані у вигляді записів, які складаються з колекцій та документів (Рисунок 3.11).

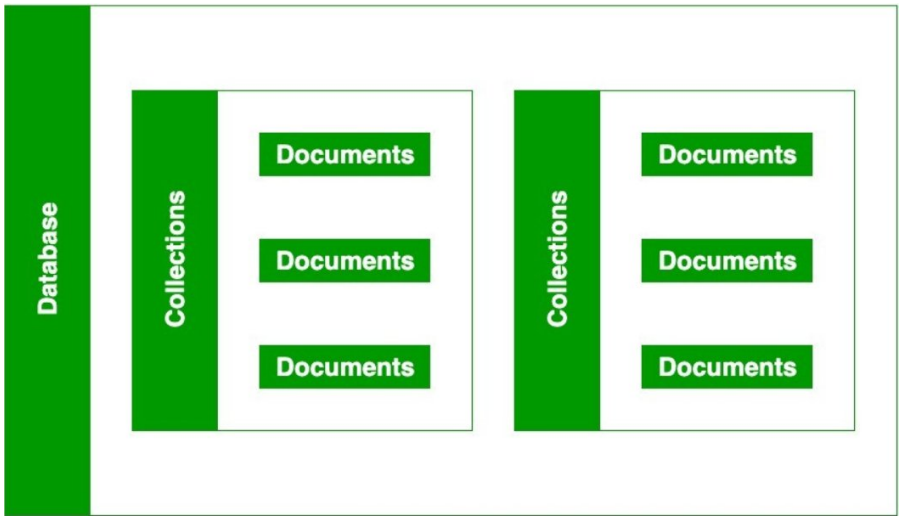


Рисунок 3.11 – Колекції та документи в «MongoDB»

Документи містять дані, які користувач хоче зберігати в базі даних MongoDB. Документи складаються з пар полів і значень. Вони є основною одиницею даних в MongoDB. Документи схожі на JavaScript Object Notation (JSON), але використовують варіант під назвою Binary JSON (BSON). Перевага використання BSON полягає в тому, що він підтримує більше типів даних. Поля в цих документах схожі на стовпці в реляційній базі даних. Значення, що містяться, можуть бути різних типів даних, включаючи інші документи, масиви і масиви документів, згідно з посібником користувача MongoDB.

Основні переваги MongoDB полягають у:

− Документоорієнтована структура. MongoDB зберігає дані у вигляді документів, які можуть бути у форматі BSON (бінарний JSON) або JSON. Кожен документ містить ключ-значення пари, і документи розташовані в колекціях.

− Гнучкість схеми. MongoDB не вимагає фіксованої схеми для даних, що дозволяє додавати та змінювати поля у документах без необхідності міграції даних.

− Горизонтальне масштабування. MongoDB підтримує горизонтальне масштабування, що дозволяє розширювати базу даних на багато серверів для обробки великих навантажень.

− Можливості запитів. MongoDB підтримує різні типи запитів, включаючи запити на знаходження, фільтрацію, агрегацію та текстовий пошук.

− Висока швидкодія. MongoDB зазвичай працює дуже швидко завдяки використанню індексів та кешування даних.

### 3.2. Опис обраного підходу до створення програмного забезпечення

Окрім вище перелічених засобів та інструментів розробки, було вирішено обрати водоспадну модель, як підхід до розробки програмного забезпечення для тестування студентів.

Водоспадна модель – це лінійний, послідовний підхід до життєвого циклу розробки програмного забезпечення, популярний в інженерії програмного забезпечення та розробці продуктів.

Водоспадна модель використовує логічну послідовність кроків життєвого циклу програмного забезпечення для проекту, подібно до того, як вода тече через край скелі. Вона встановлює чіткі кінцеві точки або цілі для кожної фази розробки. Ці кінцеві точки або цілі не можуть бути переглянуті після їх завершення [13].

Модель водоспаду продовжує використовуватися в промисловому дизайні. Її часто називають першою методологією розробки програмного

забезпечення. Модель також використовується в більш широкому сенсі як методологія управління проектами високого рівня для складних, багатогранних проектів (Рисунок 3.12).


Рисунок 3.12 – Водоспадна модель розробки ПЗ

Основні фази водоспадної моделі:

– Аналіз вимог. На початку розробки визначаються вимоги до програмного забезпечення. Здійснюється ретельний аналіз потреб користувача та визначаються функціональні вимоги.

– Проектування. Розробляється архітектура програми, обираються технології, планується структура програмного забезпечення.

– Розробка. Пишеться програмний код відповідно до розробленого проекту.

– Тестування. Після написання коду виконується тестування програми на відповідність вимогам і виправлення помилок.

– Технічна підтримка. Програма підтримується, виправляються помилки, розглядаються запити на зміни і підтримується загальна робота системи.

# 4. ПРАКТИЧНА ЧАСТИНА

## 4.1. Опис та побудова діаграми роботи системи та розгортання

Для опису роботи програмного забезпечення для тестування студентів було вирішено побудувати діаграму Прецедентів (Рисунок 4.1).

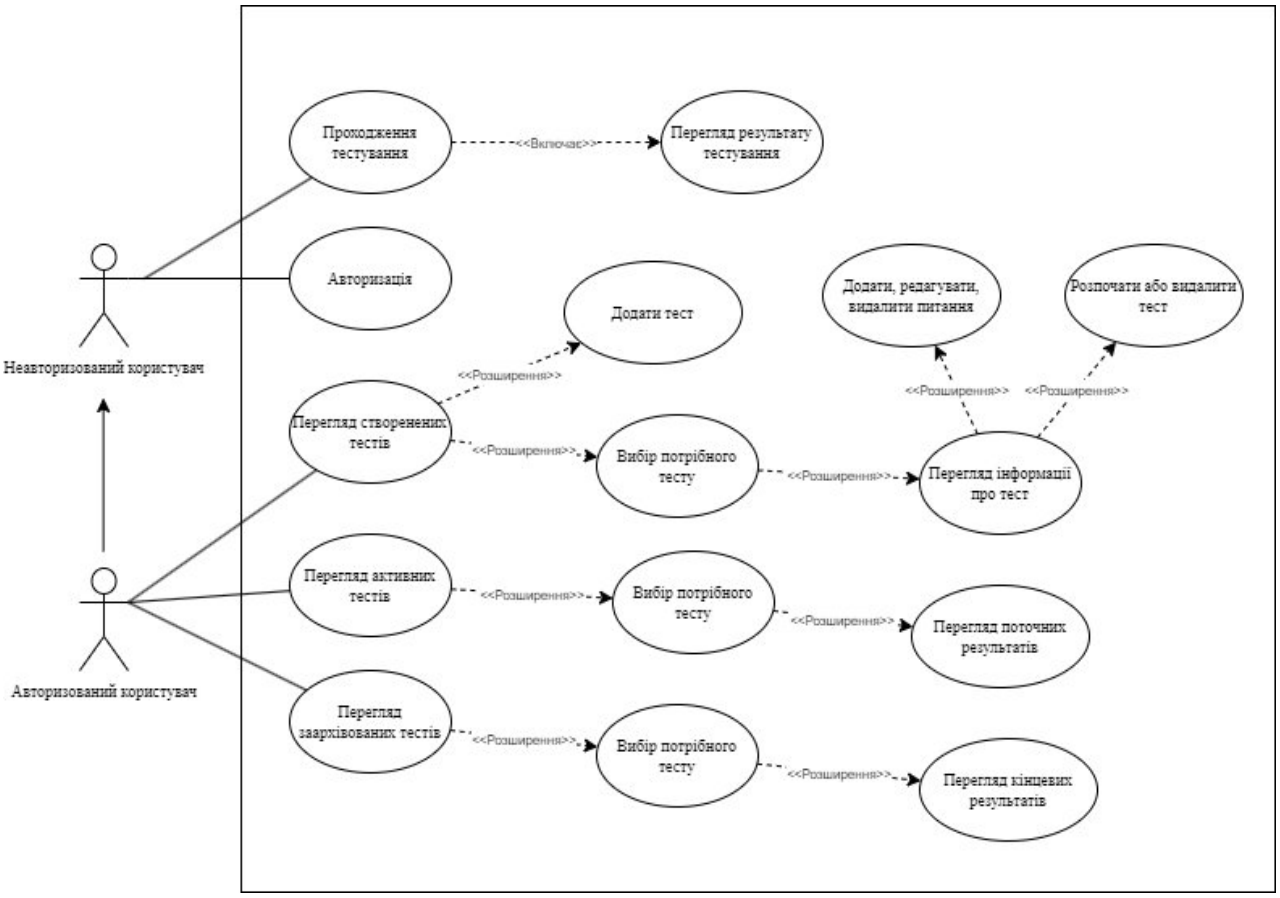


Рисунок 4.1 – Діаграма прецедентів

ПРЕЦЕДЕНТ: АВТОРИЗАЦІЯ.

Ектор: Неавторизований користувач.

Передумова: Користувач не авторизований в системі.

Післяумова: Користувач авторизований в системі.

Сценарій:

1. Користувач вводить свій email та пароль.
2. Користувач натискає кнопку «Увійти».
3. Користувач авторизований.

ПРЕЦЕДЕНТ: ПРОХОДЖЕННЯ ТЕСТУВАННЯ.

Ектор: Неавторизований користувач.

Передумова: Користувач проходить тестування.

Післяумова: Користувач пройшов тестування.

Сценарій:

1. Користувач переходить по посиланню.
2. Вводить своє «Призвіще» та «Ім’я».
3. Користувач відповідає на питання.
4. Користувач пройшов тестування

Включення: Перегляд результату тестування.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД СТВОРЕНИХ ТЕСТІВ.

Ектор: Авторизований користувач.

Передумова: Користувач авторизований у системі.

Післяумова: Користувач переглянув власні створені тести.

Сценарій:

1. Користувач переходить до вкладки «Створені тести».
2. Вибирає тест.
3. Користувач переглянув інформацію про тест.

Розширення: Додати тест, вибір потрібного тесту, перегляд інформації про тест, розпочати або видалити тест, додати питання, редагувати питання, видалити питання.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД АКТИВНИХ ТЕСТІВ.

Ектор: Авторизований користувач.

Передумова: Користувач авторизований у системі.

Післяумова: Користувач переглянув власні активні тести.

Сценарій:

1. Користувач переходить до вкладки «Активні тести».
2. Вибирає тест.
3. Користувач переглянув поточні результати тестування.

Розширення: Вибір потрібного тесту, перегляд поточних результатів.

ПРЕЦЕДЕНТ: ПЕРЕГЛЯД ЗААРХІВОВАНИХ ТЕСТІВ.

Ектор: Авторизований користувач.

Передумова: Користувач авторизований у системі.

Післяумова: Користувач переглянув заархівовані тести.

Сценарій:

1. Користувач переходить до вкладки «Заархівовані тести».
2. Вибирає тест.
3. Користувач переглянув кінцеві результати тестування.

Розширення: Вибір потрібного тесту, перегляд кінцевих результатів.

Також було вирішено створити діаграму, що пояснює алгоритм дій, які необхідно виконати для розгортання (Рисунок 4.2).

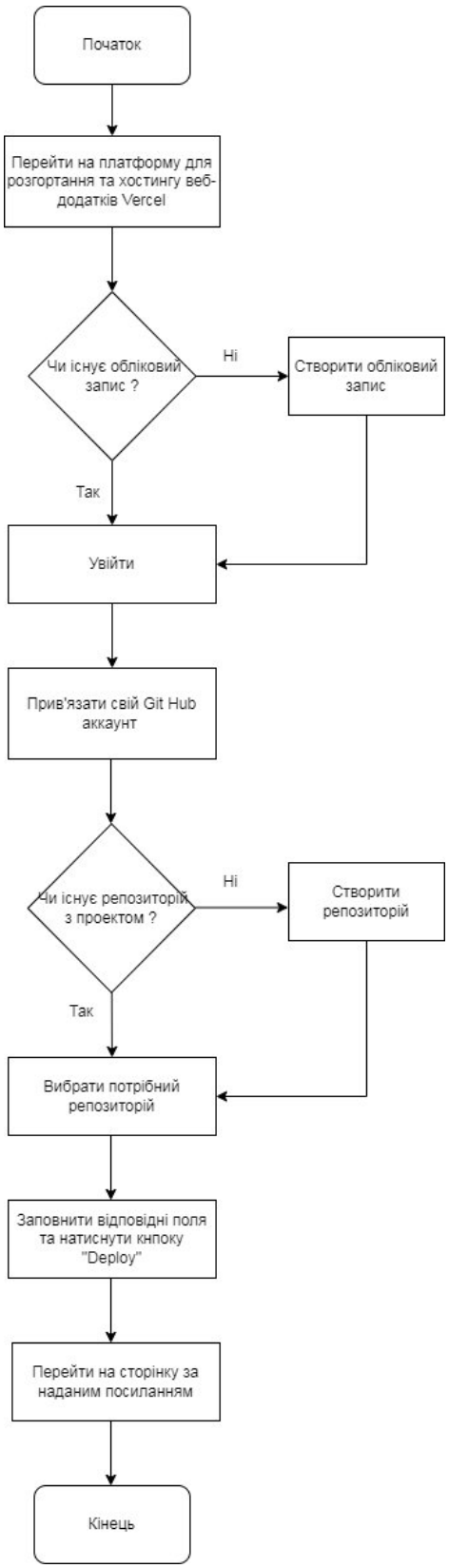


Рисунок 4.2 – Діаграма алгоритму дій розгортання веб-додатку

## 4.2. Інструкція з використання програмного забезпечення

*Створення тестів.*

1. Для того щоб створити тест потрібно увійти або зареєструватися. Для авторизації потрібно перейти на сторінку сервісу проходження тестування та заповнити відповідні поля для входу ( пошта та пароль) та натиснути кнопку «Увійти» (Рисунок 4.3).

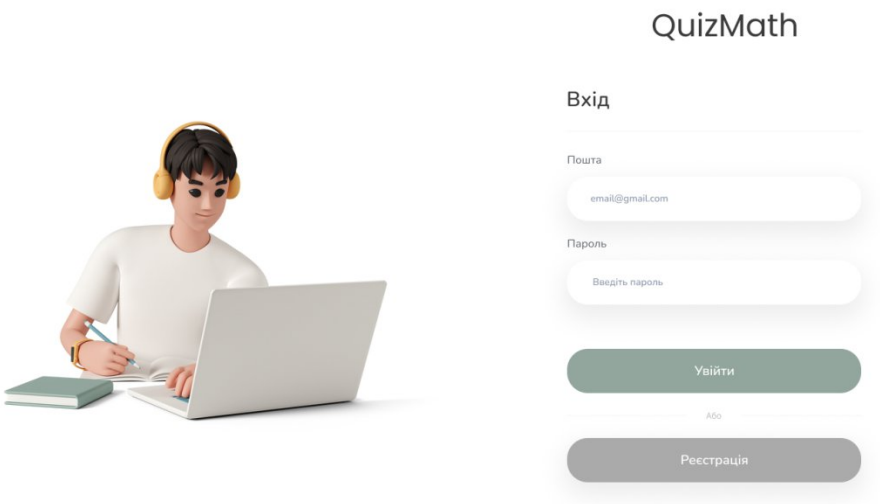


Рисунок 4.3 – Сторінка авторизації

2. Якщо користувачу потрібно зареєструватися потрібно натиснути кнопку «Реєстрація» на сторінці, та заповнити відповідні поля, а саме: ім’я, прізвище, пошта та пароль. (Рисунок 4.4).

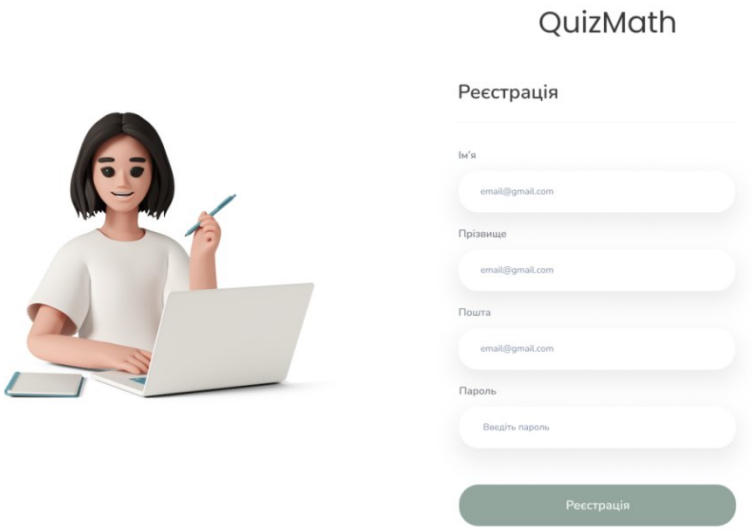


Рисунок 4.4 – Сторінка реєстрації

3. Після успішної авторизації користувач потрапляє до сторінки створення тестів. На сторінці є можливість перегляду навігаційного меню, логотипу, карток з інформацію про тест та кнопки «Додати тест »(Рисунок 4.5).

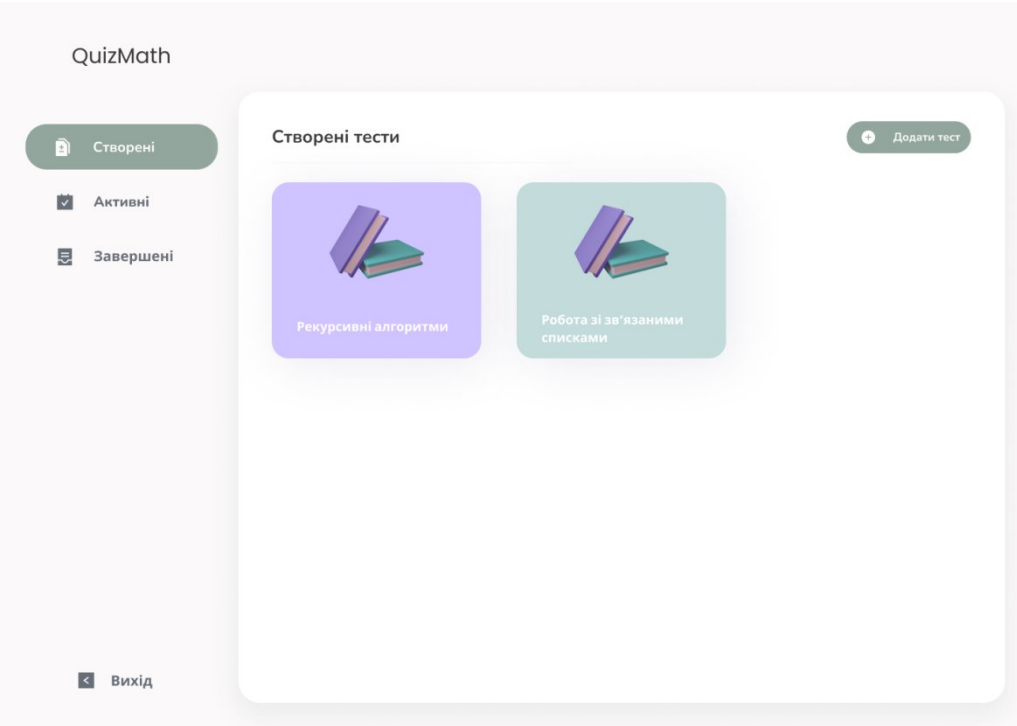


Рисунок 4.5 – Сторінка створених тестів

4. Створення тесту. Для створення тесту необхідно натиснути кнопку «Додати тест». Після чого користувача направляє на сторінку створення тесту. У відкритій сторінці необхідно заповнити поле назва тесту та створити питання з відповідями (Рисунок 4.6).

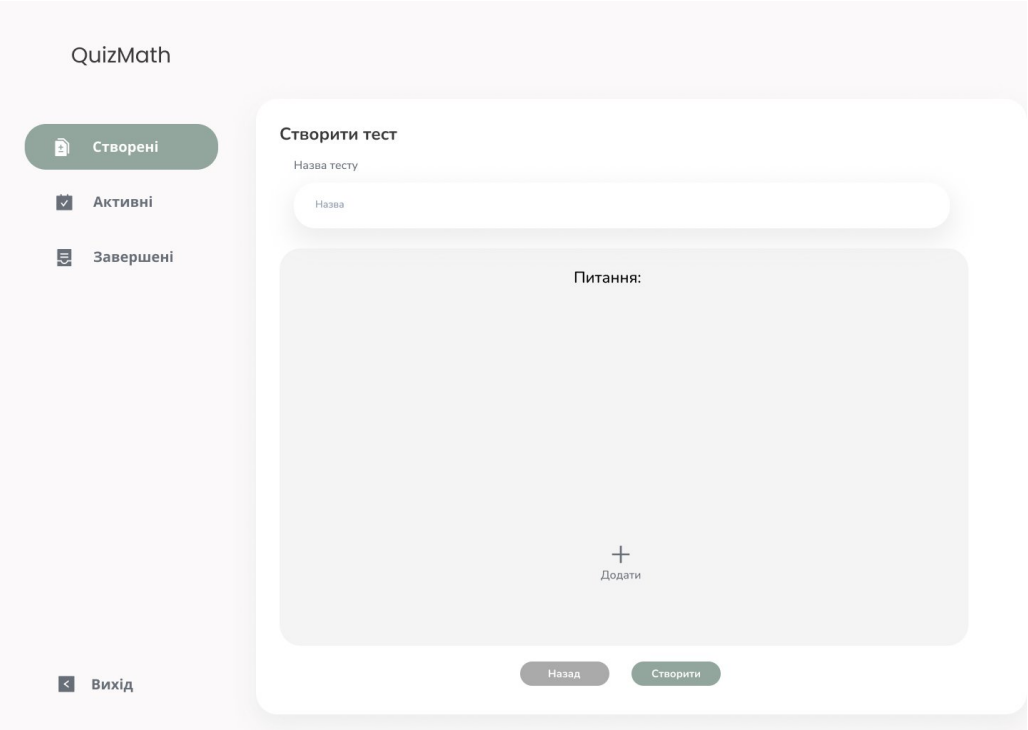


Рисунок 4.6 – Сторінка створення тесту

5. Створення питань. Для створення питань необхідно натиснути кнопку «Додати +» після чого у відкритому вікні заповнити поле текст питання та додати відповіді натисканням кнопки «Додати +». Щоб обрати правильну потрібно натиснути «✓» зліва від відповіді. Є можливість видалити правильну відповідь. Після усіх завершених дій натиснути кнопку «Додати» (Рисунок 4.7).

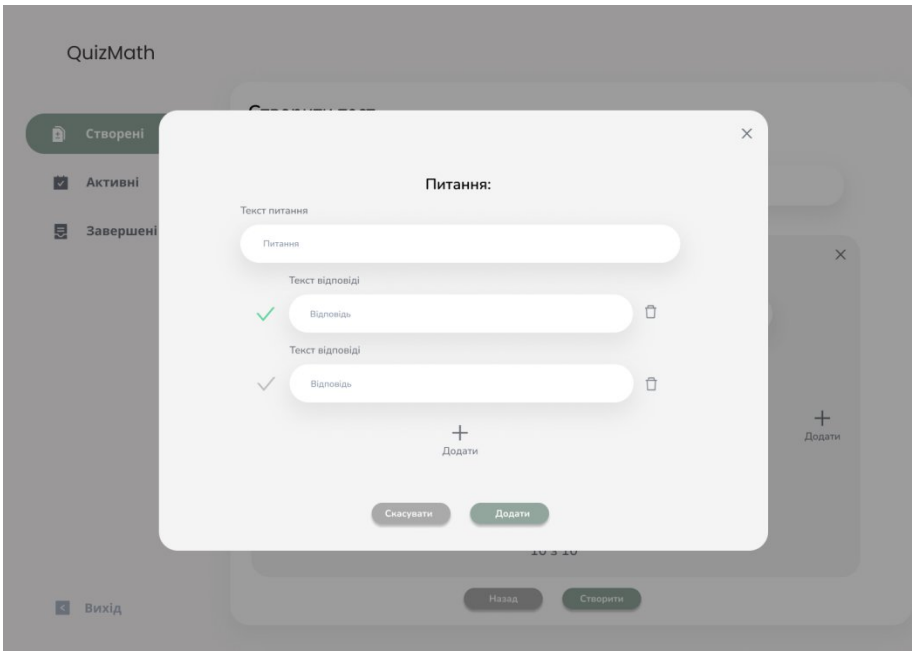


Рисунок 4.7 – Створення питання

6. Перегляд інформації про створений тест. Для перегляду інформації необхідно вибрати вкладку «Створені». У відкритій вкладці вибрати необхідний тест та натиснути на нього. Після чого відкривається сторінка з інформацією про тест, де є можливість переглянути інформацію про створений тест, відредагувати за бажанням та розпочати або видалити тест (Рисунок 4.8).

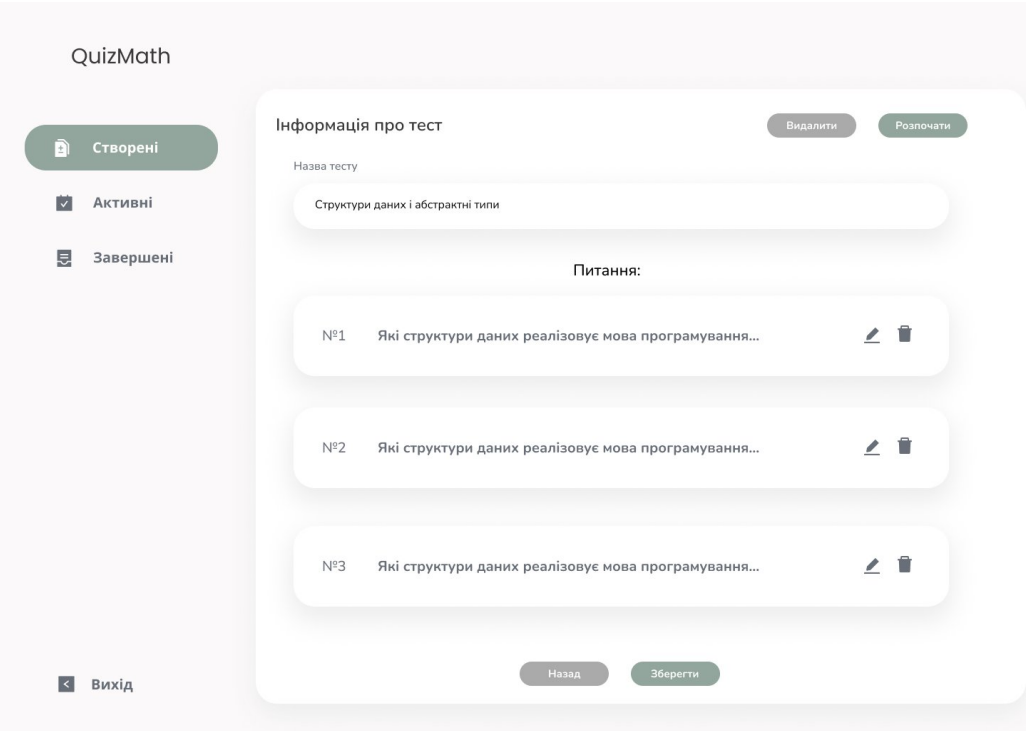


Рисунок 4.8 – Інформація про створений тест

7. Розпочати тест. Щоб розпочати тестування потрібно натиснути кнопку «Розпочати». Після цього у відкритому вікні з’являється посилання на тест у вигляді QR-коду та звичайного посилання, де є можливість його скопіювати (Рисунок 4.9).

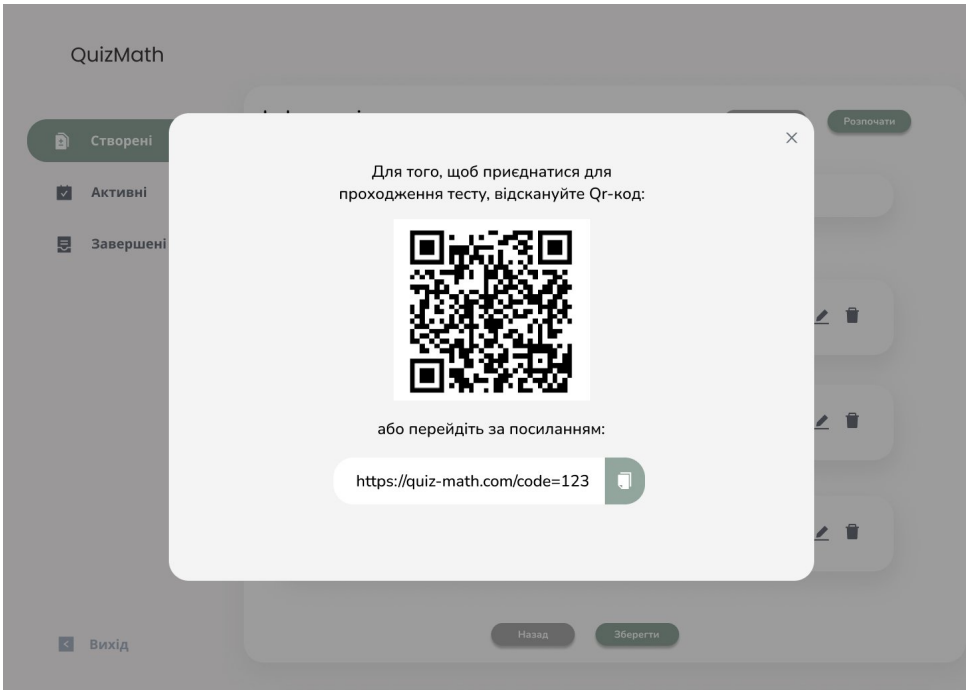


Рисунок 4.9 – Розпочати тестування

8. Перегляд результатів. Після того, як користувач розпочав тестування, тест переходить в статус активний. Його можна знайти на вкладці «Активні». Для того щоб переглянути інформацію його потрібно обрати. У відкритій сторінці є можливість перегляду результатів тестування (кількість правильних та неправильних відповідей) та зупинення тесту (Рисунок 4.10).

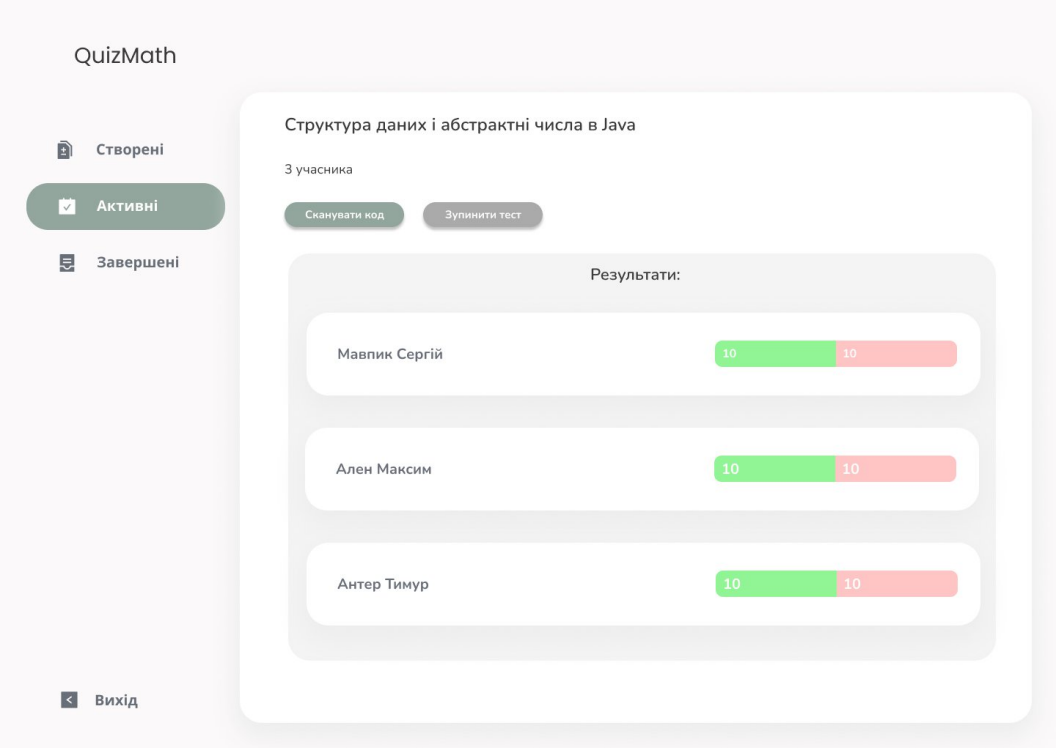


Рисунок 4.10 – Перегляд результату тестування

9. Перегляд завершених тестів. Після того, як користувач завершив тест натисканням кнопки «Зупинити тест» тест потрапляє до вкладки «Завершені». Де є можливість перегляду інформації про завершений тест (Рисунок 4.11).

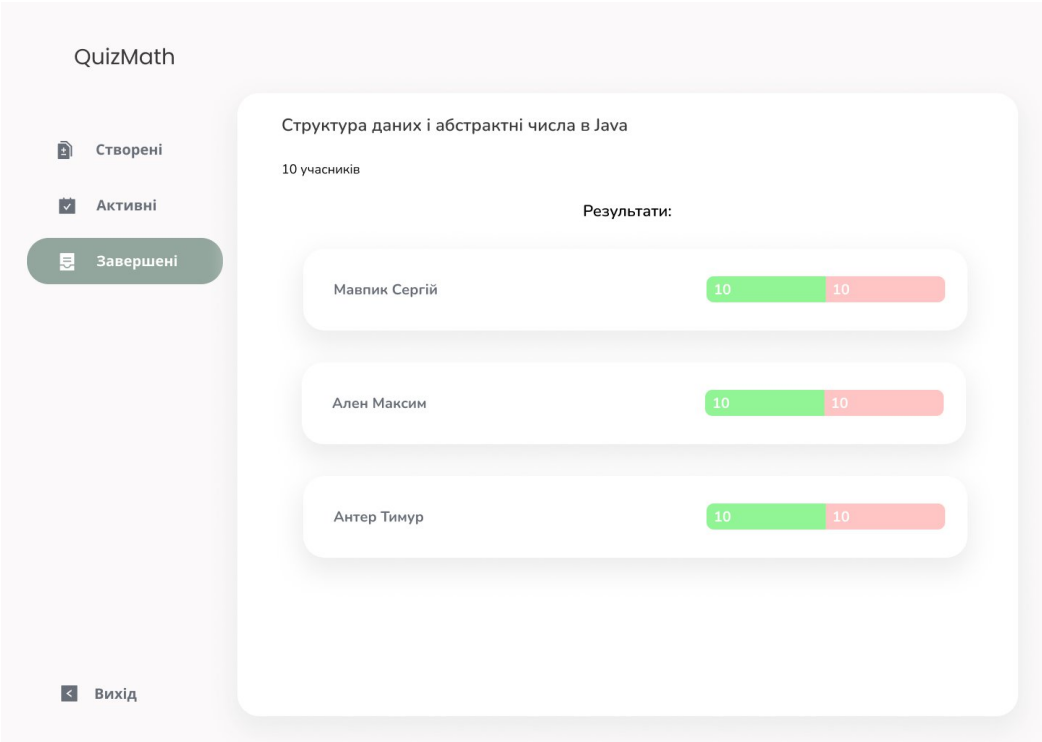


Рисунок 4.11 – Перегляд інформації про завершений тест

*Проходження тестування.*

1. Після отримання посилання для проходження тесту, користувач потрапляє на сторінку, де повинен заповнити поле «Прізвище та ім’я» та натиснути кнопку «Приєднатися» (Рисунок 4.12).

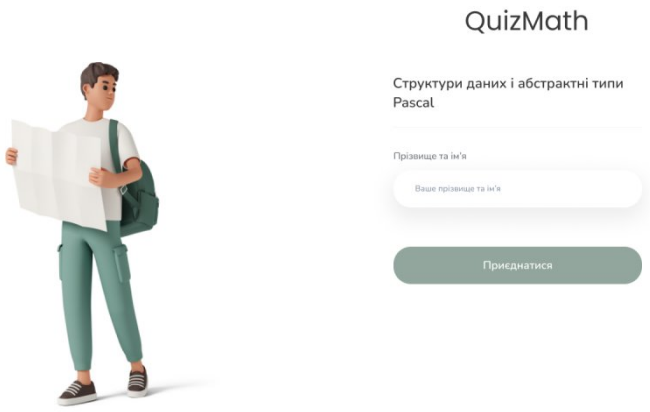


Рисунок 4.12 – Приєднатися до тесту

2. Після приєднання, користувач потрапляє на сторінку, де є можливість перегляду запитання та вибір однієї правильної відповіді (Рисунок 4.13). Якщо користувач відповів правильно, на екрані з’являється повідомлення «Відповідь правильна» (Рисунок 4.14), в іншому випадку навпаки (Рисунок 4.15).

Рисунок 4.13 – Проходження тестування

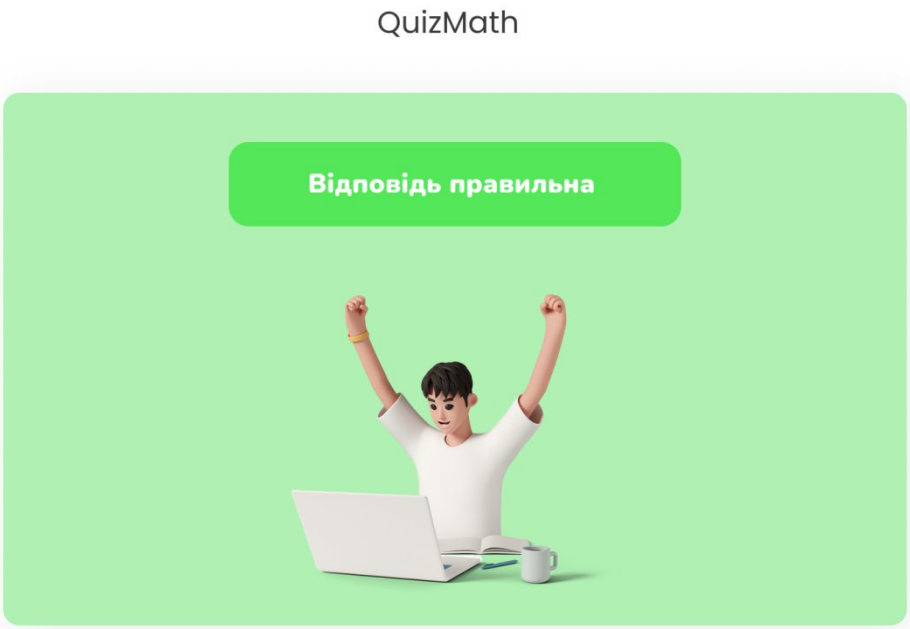


Рисунок 4.14 – Відповідь правильна

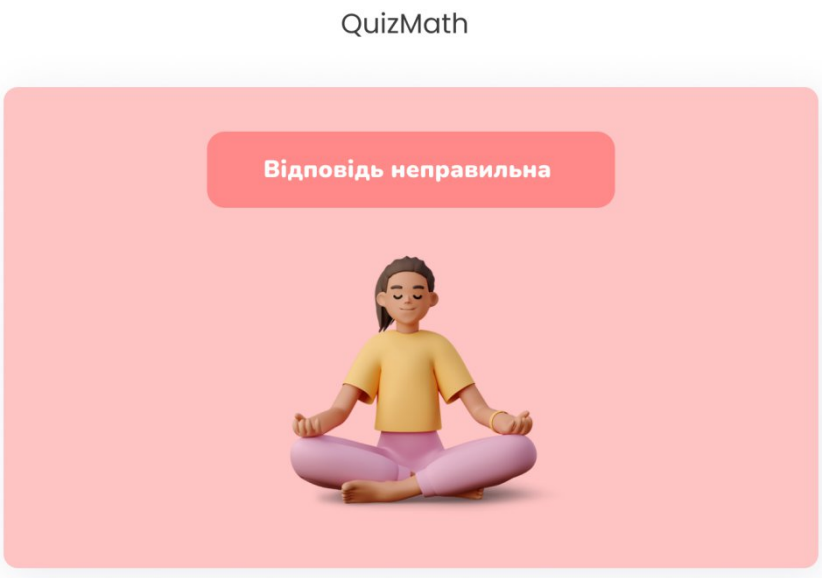


Рисунок 4.15 – Відповідь неправильна

3. Перегляд результату тесту. Після того, як користувач відповів на остання запитання у нього є можливість перегляду результату тестування (Рисунок 4.16).

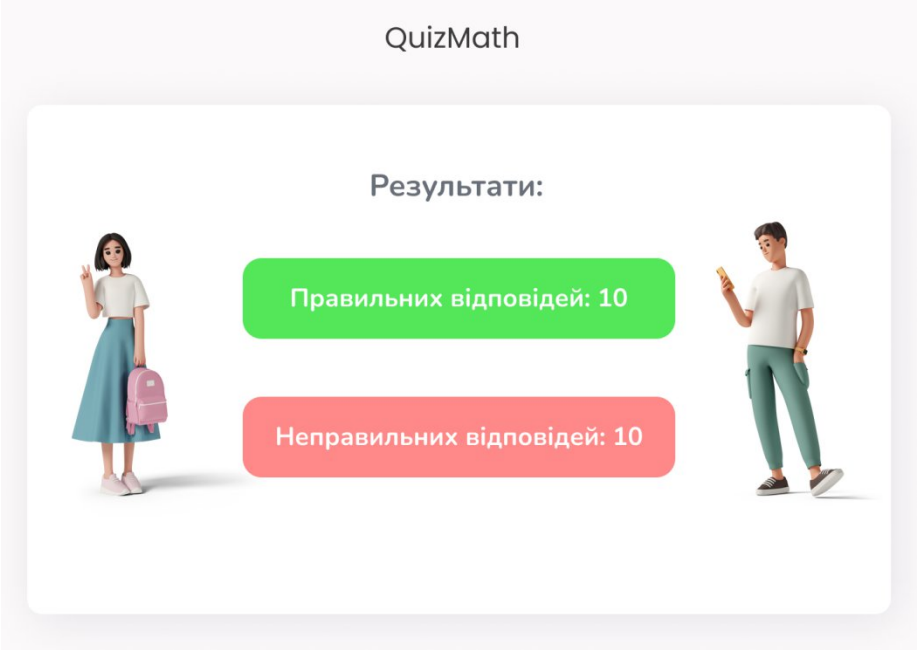


Рисунок 4.16 – Перегляд результату тестування

# ВИСНОВКИ

В умовах сучасності, з розвитком технологій впровадження дистанційного навчання стає все більш поширеним, адже воно дозволяє спростити та підлаштувати процес навчання під власний темп, що збільшує його ефективність, а також надає можливість автоматизованого оцінювання за допомогою різних інструментів серед яких онлайн-тестування займає досить важливу роль.

Згідно завдання було створено програмне забезпечення для тестування студентів з дисципліни «Алгоритми і структури даних».

Основні результати роботи:

- визначено актуальність теми кваліфікаційної роботи;
- сформульовані основні вимоги до програмного забезпечення;
- проведено огляд систем аналогічного призначення, виділені їх основні переваги;
- виконано опис проектних рішень та інструментів до розробки програмного забезпечення;
- обрано методологію створення програмного забезпечення;
- побудовано діаграму прецедентів та діаграму алгоритму при розгортанні;
- розроблено програмне забезпечення для тестування студентів, реалізовано всю функціональність, що була описана у вимогах;
- складено інструкцію з користуванням.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Google Forms (Гугл Форми): ТОП 10 лайфхаків для отримання максимального ефекту від використання. URL: https://fotc.com/ua/blog/google-forms-10-lajfhakiv-dlya-maksimalnogo-efektu.
2. Онлайн-платформа навчання Kahoot. URL: https://samoosvita.in.ua/onlayn-platforma-navchannya-kahoot/.
3. Створення і використання тестів, вікторин при вивчені різних предметів на базі онлайн конструктора Quizizz. URL: https://karpecira.blogspot.com/2021/02/quizizz.html?m=1.
4. Навчання під час війни: як організувати процес з Classtime. URL: https://znayshov.com/News/Details/navchannia_pid_chas_viiny_Yak_orhanizuvaty_protses_z_classtime.
5. Why You Should Use TypeScript for Developing Web Applications. URL: https://dzone.com/articles/what-is-typescript-and-why-use-it.
6. Why Next.js 13 is a Game-Changer. URL: https://blog.bitsrc.io/why-next-js-13-is-a-game-changer-2167658d9de2.
7. A Beginner’s Guide to Sass. URL: https://www.codecademy.com/resources/blog/what-is-sass/.
8. What is React.js? (Uses, Examples, & More). URL: https://blog.hubspot.com/website/react-js.
9. What is Zod? URL: https://www.educative.io/answers/what-is-zod.
10. What is Visual Studio Code? Microsoft’s extensible code editor. URL: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html.
11. What is Git and Why Should You Use it? URL: https://www.nobledesktop.com/learn/git/what-is-git.
12. What is MongoDB? Features and how it works. URL: https://www.techtarget.com/searchdatamanagement /definition/MongoDB.

13. What is the waterfall model? URL: https://www.techtarget.com/searchsoftwarequality /definition/waterfall-model.
14. Welcome to the Next.js documentation! URL: https://nextjs.org/docs.
15. Tochukwu John. Sharing Layouts with Next.js 13's app dir. URL: https://blog.bitsrc.io/sharing-layouts-with-nextjs-13-app-dir-6b64e3242d82.
16. Joel Chi. Next.js – A brief overview. URL: https://levelup.gitconnected.com/next-js-a-brief-overview-78ede74a22b9.
17. Emmanuel Odioko. Using Next.js Route Handlers. URL: https://blog.logrocket.com/using-next-js-route-handlers.
18. Alen Ajam. How to Structure Your Next.js App With the New App Router. URL: https://betterprogramming.pub/how-to-structure-your-next-js-app-with-the-new-app-router-61bf2bf5a20d.
19. Samer Buna. All the fundamental React.js concepts, jammed into this one article. URL: https://www.freecodecamp.org/news/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2/.
20. Noring C. React Book, your beginner guide to React. 2021. https://softchris.github.io/books/react.
21. Copes F. The Next.js Handbook. 2022, 102. https://tuto-computer.com/others/1748-the-nextjs-handbook.html.

# ДОДАТОК А. ВИХІДНІ КОДИ

ГОЛОВНИЙ ФАЙЛ

```
import { Nunito } from 'next/font/google';
import { getServerSession } from 'next-auth';
import AuthProvider from '@/components/AuthProvider/AuthProvider';
import { authOptions } from './api/auth/[...nextauth]/route'
import 'react-responsive-modal/styles.css';
import './../styles/main.scss';
export const metadata = {
title: 'Quiz math',
description: 'With love',
};
const nunito = Nunito({
subsets: ['cyrillic', 'latin'],
display: 'swap',
});
export default async function RootLayout({ children }: { children:
React.ReactNode }) {
const session = await getServerSession(authOptions);
return (
<html lang='uk-UA' className={nunito.className}>
<body>
<AuthProvider session={session}>
<main>{children}</main>
</AuthProvider>
</body>
</html>
);
```

```
}
```

СТОРІНКА СТВОРЕНІ ТЕСТИ

```
import Link from 'next/link';
import { AiFillPlusCircle } from 'react-icons/ai';
import Card from '@/components/card/Card';
import connect from '@/database/connection';
import TestTemplate from '@/database/models/testTemplate.model';
export default async function Created() {
await connect();
const tests = await TestTemplate.find();
const listCard = tests.map((test) => (
<Card
key={test._id}
id={test._id.toString()}
name={test.title}
color={test.color}
route='created'
/>
));
return (
<>
<div className='page__title-container'>
<div className='page__title'>
<span>Створені тести</span>
</div>
<Link href='created/add' className='btn primary'>
<AiFillPlusCircle />
Додати тест
</Link>
```

```
</div>
<div className='page__line' />
<div className='page__card-container'>{listCard}</div>
</>
);
}
```

СТОРІНКА АВТОРИЗАЦІЇ В СИСТЕМУ

```
'use client';
import Image from 'next/image';
import '@/styles/pages/_auth.scss';
import { usePathname } from 'next/navigation';
import { useSession } from 'next-auth/react';
export default function AuthTemplate({ children }: { children: React.ReactNode }) {
const pathname = usePathname();
return (
<div className='auth-page'>
<div className='auth-page__left'>
<div className='auth-page__left-container'>
<Image
className='left-container__image'
src={pathname === '/login' ? '/manLaptop.png' : '/womanLaptop.png'}
priority
width={640}
height={450}
alt=''
/>
</div>
</div>
<div className='auth-page__right'>
```

```
<div className='auth-page__right-container'>
<h1 className='right-container__logo'>QuizMath</h1>
{children}
</div>
</div>
</div>
);
}
```

СТВОРЕННЯ ТЕСТУ

```
'use client';
import { useState } from 'react';
import { Toaster } from 'react-hot-toast';
import { Controller, useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { AiOutlinePlus } from 'react-icons/ai';
import Button from '@/components/ui/Button';
import Input from '@/components/ui/Input';
import { TestTemplateInput } from '@/database/models/testTemplate.model';
import { QuestionInput } from '@/database/models/question.model';
import { TestTemplateApiService } from '@/lib/api/services/testTemplate.service';
import QuestionModal from '@/components/modals/question/Question';
import Question from '@/components/question/Question';
import { testSchema } from './schemas';
import { TestSchemaType } from './types';
interface Props extends Partial<TestTemplateInput> {}
const TestForm = (props: Props) => {
const { title } = props;
let { questions } = props;
const [showModal, setShowModal] = useState(false);
```

```
const [modalMode, setModalMode] = useState<'create' | 'edit'>('create');
const [selectedQuestion, setSelectedQuestion] = useState<QuestionInput>();
const {
control,
getValues,
setValue,
handleSubmit,
formState: { errors, isSubmitting },
} = useForm<TestSchemaType>({
mode: 'onSubmit',
reValidateMode: 'onChange',
resolver: zodResolver(testSchema),
defaultValues: { title, questions },
});
const handleShow = () => setShowModal(true);
const handleClose = () => setShowModal(false);
const handleCloseAnimationEnd = () => setSelectedQuestion(undefined);
const handleQuestionAddStart = () => {
setModalMode('create');
handleShow();
};
const handleQuestionEditStart = (index: number) => {
const questions = getValues('questions');
if (!questions  !questions.length) return;
setSelectedQuestion(questions[index]);
setModalMode('edit');
handleShow();
};
const handleQuestionAdd = (question: QuestionInput) => {
const questions = getValues('questions')  [];
```

```
questions.push(question);
setValue('questions', questions);
};
const handleCreate = async (data: TestSchemaType) => {
await TestTemplateApiService.createOne({ ...data, color: '#ffffff' });
};
const renderQuestionsList = () => {
const questions = getValues('questions') || [];
return questions?.map(({ text }, index) => (
<Question
key={index}
index={index + 1}
text={text}
onStartEdit={() => handleQuestionEditStart(index)}
/>
));
};
return (
<>
<Toaster />
<QuestionModal
{...(selectedQuestion && selectedQuestion)}
mode={modalMode}
show={showModal}
onSubmit={handleQuestionAdd}
onClose={handleClose}
onCloseAnimationEnd={handleCloseAnimationEnd}
/>
<form onSubmit={handleSubmit(handleCreate)}>
<div className='page__input-quiz'>
```

```
<div className='form-input'>
<Controller
control={control}
name='title'
defaultValue=''
render={({ field: { onChange, onBlur, value } }) => (
<Input
name='title'
label='Назва тесту'
placeholder='Назва'
onChange={onChange}
onBlur={onBlur}
value={value}
disabled={isSubmitting}
/>
)}
/>
{errors.title && <p className='form-error'>{errors.title.message}</p>}
</div>
</div>
<div className='page__question-container'>
<div className='questions'>
<p className='questions__title'>Питання:</p>
<div className='questions__list'>{renderQuestionsList()}</div>
<div className='questions__button-container'>
<div className='questions__button' onClick={handleQuestionAddStart}>
<AiOutlinePlus size={22} />
<p>Додати питання</p>
</div>
</div>
```

```
</div>
</div>
<div className='page__button-footer'>
<Button color='secondary'>Назад</Button>
<Button type='submit' color='primary'>
Створити
</Button>
</div>
</form>
</>
);
};
export default TestForm;
```

СТВОРЕННЯ ЗАПИТАННЯ З ВІДПОВІДЯМИ

```
'use client';
import { Modal } from 'react-responsive-modal';
import { AiOutlineClose, AiOutlinePlus } from 'react-icons/ai';
import { Controller, useFieldArray, useForm } from 'react-hook-form';
import Input from '@/components/ui/Input';
import Answer from '@/components/answer/Answer';
import Button from '@/components/ui/Button';
import { QuestionInput } from '@/database/models/question.model';
import { AnswerInput } from '@/database/models/answer.model';
import { questionSchema } from './schemas';
import { zodResolver } from '@hookform/resolvers/zod';
import { QuestionSchemaType } from './types';
import { useEffect } from 'react';
const fieldArrayName = 'answers';
const fieldArrayInputDefaultValues: AnswerInput = { text: '', isCorrect: false };
```

```
const fieldArrayDefaultValues: AnswerInput[] = new
Array(2).fill(fieldArrayInputDefaultValues);
interface Props extends Partial<QuestionInput> {
mode: 'create' | 'edit';
show: boolean;
onSubmit: (question: QuestionInput) => void;
onClose: () => void;
onCloseAnimationEnd: () => void;
}
const QuestionModal = (props: Props) => {
const { text, answers, mode, show, onSubmit, onClose, onCloseAnimationEnd } =
props;
const {
control,
getValues,
setValue,
handleSubmit,
reset,
formState: { errors, isSubmitting },
} = useForm<QuestionSchemaType>({
mode: 'onSubmit',
reValidateMode: 'onChange',
resolver: zodResolver(questionSchema),
defaultValues: {
text,
[fieldArrayName]: answers?.length ? answers : fieldArrayDefaultValues,},
});
const { fields, append, remove } = useFieldArray({ control, name:
fieldArrayName });
const handleQuestionSubmit = (data: QuestionSchemaType) => {
```

```
onSubmit(data);
onClose();
};
const handleAnimationEnd = () => {
if (!show) {
reset({ text: '', answers: fieldArrayDefaultValues });
onCloseAnimationEnd();}
};
const removePreviousCorrectAnswer = () => {
const answers = getValues('answers');
setValue(
'answers',
answers.map(({ text }) => ({ text, isCorrect: false })),
);
};
const removeAnswerField = (index: number) => {
if (fields.length > 2) remove(index);
};
useEffect(() => {
if (text && answers?.length) reset({ text, answers });
}, [reset, text, answers]);
const answersInputs = fields.map((field, index) => {
return (
<Answer
key={field.id}
index={index}
control={control}
value={field}
error={errors.answers?.[index]?.text?.message}
remove={removeAnswerField}
```

```
onCorrectCheck={removePreviousCorrectAnswer}
/>
);
});
return (
<Modal
open={show}
onClose={onClose}
onAnimationEnd={handleAnimationEnd}
center
classNames={{ modal: 'quiz-modal' }}
focusTrapped={false}
closeIcon={<AiOutlineClose size={22} />}
>
<div className='question-modal'>
<p className='question-modal__title'>Питання:</p>
<form onSubmit={handleSubmit(handleQuestionSubmit)}>
<div className='form-input'>
<Controller
control={control}
name='text'
defaultValue=''
render={({ field: { onChange, onBlur, value } }) => (
<Input
name='text'
label='Текст питання'
placeholder='Питання'
onChange={onChange}
onBlur={onBlur}
value={value}
```

```
disabled={isSubmitting}
/>
)}
/>
{errors.text && <p className='form-error'>{errors.text.message}</p>}
</div>
<div className='question-modal__answer-container'>{answersInputs}</div>
<div className='question-modal__error-container'>
{errors.answers && <p className='form-error'>{errors.answers.message}</p>}
</div>
<div className='questions__button-container'>
<div
className='questions__button'
onClick={() => append(fieldArrayInputDefaultValues)}
>
<AiOutlinePlus size={22} />
<p>Додати відповідь</p>
</div>
</div>
<div className='question-modal__button-footer'>
<Button onClick={onClose} color='secondary'>
Назад
</Button>
<Button type='submit' color='primary'>
{mode === 'create' ? 'Додати' : 'Редагувати'}
</Button>
</div>
</form>
</div>
</Modal>
```

```
);
};
export default QuestionModal;
```

ВІКНО З ПОСИЛАННЯМ ТА QR-КОДОМ НА ТЕСТ

```
"use client";
import React from "react";
import { Modal } from "react-responsive-modal";
import { AiOutlineClose, AiFillCopy } from "react-icons/ai";
import { QRCodeSVG } from "qrcode.react";
import Button from "@/components/ui/Button";
interface IProps {
onStart: boolean;
handleClose: () => void;
}
const QuizInfo = (props: IProps) => {
const { onStart, handleClose } = props;
return (
<Modal
open={onStart}
onClose={handleClose}
center
classNames={{ modal: "quiz-modal" }}
focusTrapped={false}
closeIcon={<AiOutlineClose size={22} />}
>
<div className='question-modal'>
<p className='question-modal__title'>
Для того, щоб приєднатися для проходження тесту, відскануйте Qr-код:
</p>
```

```
<div className='question-modal__code'>
<QRCodeSVG value='65045fac163735c3fa3df29c' size={250} />
</div>
<p className='question-modal__title'>або перейдіть за посиланням:</p>
<div className='question-modal__form'>
<div className='question-modal__input'>
</div>
<Button
color='primary'
form='rounded'
onClick={() => {
navigator.clipboard.writeText("");
}}
>
<AiFillCopy size={25} />
</Button>
</div>
</div>
</Modal>
);
};
export default QuizInfo;
```

НАВІГАЦІЯ

```
'use client';
import React from 'react';
import Link from 'next/link';
import { usePathname } from 'next/navigation';
import { signOut } from 'next-auth/react';
import { BiSolidLogOut } from 'react-icons/bi';
```

```
import { sidebarData } from './constants';

const Sidebar = () => {
const currentRoute = usePathname();
const listSidebar = sidebarData.map((link) => (
<li key={link.id}>
<Link
className={
currentRoute.startsWith(link.url)
? 'sidebar__item-container sidebar__active'
: 'sidebar__item-container'
}
href={link.url}
>
<div className='sidebar__icon'>{link.icon}</div>
<span className='sidebar__title'>{link.title}</span>
</Link>
</li>
));
return (
<aside className='sidebar'>
<div className='sidebar__container'>
<div className='sidebar__header'>
<a className='sidebar__logo'>QuizMath</a>
<ul className='sidebar__list'>{listSidebar}</ul>
</div>
<div className='sidebar__footer'>
<button
onClick={() => signOut({ callbackUrl: '/login' })}
className='sidebar__footer-button'
```

```
>
<div className='sidebar__icon'>
<BiSolidLogOut size={25} />
</div>
<span className='sidebar__title'>Вихід</span>
</button>
</div>
</div>
</aside>
);
};
export default Sidebar;
```

КНОПКА

```
import React from "react";
interface Props extends React.ButtonHTMLAttributes<HTMLButtonElement> {
children: React.ReactNode;
color: "primary" | "secondary";
form?: "rounded";
}
export default function Button({ children, form, color, ...attributes }: Props) {
return (
<button type='button' className={btn ${color} ${form}} {...attributes}>
{children}
</button>
);
}
```

ТЕКСТОВЕ ПОЛЕ

```
import React from "react";
```

```
interface Props extends React.InputHTMLAttributes<HTMLInputElement> {
name: string;
label?: string;
}
export default function Input({ name, label, ...attributes }: Props) {
return (
<div className='input-wrapper'>
<label className='input-wrapper__title' htmlFor={name}>
{label}
</label>
<input className='input-wrapper__input' id={name} {...attributes}></input>
</div>
);
}
```