Галицький фаховий коледж імені В’ячеслава Чорновола
відділення комп’ютерних технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділення
комп’ютерних технологій
Наталія СТЕФУРАК /_____/
підпис
«___» _____ 202__ р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
освітньо-професійного ступеня «фаховий молодший бакалавр»
зі спеціальності 122 «Комп’ютерні науки»

на тему: «Вебсервіс для вивчення англійської мови учнями початкової школи»

Студент групи КН-41	Назарій МОСКАЛЮК	_____ (підпис)
Керівник роботи	Наталія КУЛЬЧИНСЬКА	_____ (підпис)
Консультанти:		
з техніко-економічного обґрунтування	Любов МЕЛЕНЧУК	_____ (підпис)
нормоконтролер	Надія ГАВРИШКІВ	_____ (підпис)

Тернопіль – 2024

Галицький фаховий коледж імені В’ячеслава Чорновола
відділення комп’ютерних технологій
циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділення
комп’ютерних технологій
Наталія СТЕФУРАК /_____/
підпис
«___» _____ 202__ р.

ЗАВДАННЯ
на кваліфікаційну роботу
на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»
студенту Москалюку Назарію Ярославовичу
(прізвище, ім’я та по-батькові студента)

1. Тема роботи _____
_____
_____
затверджена наказом по коледжу від “___”_____202_ р., №_____

2. Термін здачі студентом завершеної роботи “___” _____202_ р.
3. Вихідні дані до роботи _____
_____
_____

4. Перелік питань, які повинні бути розроблені:
а) основна частина _____
_____
б) техніко-економічне обгрунтування _____
_____
_____
_____
5. Перелік графічного матеріалу _____
_____
_____

6. Консультанти роботи: ___

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
З техніко-економічного обґрунтування	_____ _____ (вчена ступень, звання П.І.Б консультанта)		

КАЛЕНДАРНИЙ ПЛАН

Виконання кваліфікаційної роботи

№ п/п	Найменування етапу	Терміни	
		початку	Завершення

7. Дата видачі завдання “___” _____ 202_ р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Кваліфікаційна робота на тему «Вебсервіс для вивчення англійської мови учнями початкової школи» складається з 106 сторінок, 66 ілюстрацій, 4 таблиць, 2 додатків та 8 джерел у переліку посилань.

Використовуватимуться різні програмні забезпечення, що дозволяють створити базу даних, та під’єднавши, реалізувати можливість проходження тестувань.

Перед реалізацією, буде проведено аналіз популярних існуючих рішень, що дозволило формалізувати вимоги до проєкту. Вибрані технології не базувались на цьому аналізі, так як в більшості не мають відмінностей для кінцевого користувача. Вебсервіс може існувати лише при підтримці онлайн хостингу як сайту так і бази даних.

Використовуючи вибрані технології, буде реалізовано методи, що дозволяють взаємодіяти з даними в таблиці простому користувачу. При подальшій модифікації даного проєкту, рекомендується додати ширший обсяг можливих типів тестів, які користувачі можуть створити. Створена вебсистема функціонує без проблем, допомагаючи школярам набувати нові знання.

Abstract

The qualification work on the topic «Вебсервіс для вивчення англійської мови учнями початкової школи» consists of 106 pages, 66 illustrations, 4 tables, 2 appendices and 8 sources in the list of references.

Various software will be used to create a database, and by connecting it, implement the possibility of passing tests.

Before implementation, an analysis of popular existing solutions will be conducted, which allowed to formalize the requirements for the project. The selected technologies were not based on this analysis, as most do not make a difference to the end user. A web service can exist only with the support of online hosting for both the site and the database.

Using the selected technologies, methods will be implemented that allow a simple user to interact with the data in the table. When further modifying this project, it is recommended to add a wider range of possible types of tests that users can create. The created web system functions without problems, helping schoolchildren acquire new knowledge.

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ 7
1.1 Дослідження предметної області 7
1.2 Обґрунтування вибору теми 8
1.3 Аналіз наявних рішень 9
1.4 Аналіз вимог та постановка завдання 14
2 ПРОЄКТУВАННЯ СИСТЕМИ 17
2.1 Формалізація вимог 17
2.2 Проєктування структури 19
2.3 Аналіз технологій реалізацій 22
2.4 Проєктування інтерфейсу 24
2.5 Проєктування бази даних 31
3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ 34
3.1 Реалізація інтерфейсу 34
3.2 Реалізація функціоналу 67
3.3 Реалізація бази даних 76
3.4 Тестування системи 80
4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ 84
4.1 Аналіз ринку збуту продукту 84
4.2 Розрахунок витрат на проектування 84
4.3 Обґрунтування необхідності розробки 86
ВИСНОВКИ 88
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 89
ДОДАТКИ 90

ВСТУП

Програмне забезпечення, яке покращує процес навчання, стає все більш популярним серед педагогів. Сучасний освітній процес використовує ці інструменти, щоб дозволити вчителям ефективніше організовувати уроки, а учням краще навчатися. Вивчення англійської мови є важливим для загального рівня знань учня. Навчання має бути максимально ефективним, адже знання іноземної мови відкриває багато можливостей для подальшої освіти та професійного розвитку.

Для процесу викладання потрібні інноваційні рішення. Традиційні методи навчання не відповідають потребам сучасних учнів, які використовують інтерактивні та цифрові технології.

Було створено багато платформ, щоб задовольнити потреби вчителів, які можуть використовувати ці платформи для створення навчальних ресурсів. Деякі платформи зосереджені на інтерактивних курсах, деякі на самонавчанні, а деякі на тестуванні та оцінці знань.

Метою кваліфікаційної роботи є розробка веб-системи, яка дозволяє вчителям створювати освітній контент для навчання учнів молодшого шкільного віку англійської мови. Система дасть можливість учням вивчати у цікавій інтерактивній формі матеріал відповідно до навчальної програми та пройти тестування з отриманням результатів, що допоможе оцінити свої знання та виявити прогалини в навчанні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Дослідження предметної області

Англійська мова дуже важлива, все більше батьків починають віддавати своїх дітей на курси іноземних мов ще до того, як вони підуть до школи. Дійсно, чим раніше діти починають вчитися, тим краще їм це вдається. У підлітковому віці це дозволяє розуміти світові тренди та спілкуватися з однолітками з інших країн, або вступити в омріяний вищий навчальний заклад.

Створення вебсервісу для вивчення основ англійської мови, спрямований саме на учнів початкової школи є важливою ініціативою. Він може бути використаний для:

– покращення якості освіти шляхом заохочення до вивчення іноземних мов в початкових класах;

– підвищення середнього рівня знань школяра в Україні;

– розвиток навичок читання та письма;

– використання ігрових елементів для стимулювання інтересу учнів та покращити процес навчання;

– підготовка до здачі зовнішнього незалежного оцінювання або тестів отримання сертифікатів яких, підтверджує володіння мовою на певному рівні;

– вебсервіс може служити додатковим ресурсом для вчителів, які можуть використовувати його для підтримки навчального процесу в класі.

Саме тому, створення вебсервісу з вивчення англійської мови є важливим етапом для того, щоб показати батькам та їх дітям, які нові можливості відкривають знання іноземних мов.

1.2 Обґрунтування вибору теми

У сучасному глобалізованому світі англійська мова стала необхідним інструментом для спілкування, навчання, розвитку кар'єри та культурного розуміння. Англійська мова вибрана зважаючи на свою всесвітню популярність та важливість. На рисунку 1.1, можна побачити статистику в якій 51% українців зазначили, що мають деякі знання з англійської мови. Однак, як зазначають соціологи, детальніший аналіз показує, що тільки 23% з них можуть читати, писати та спілкуватися цією мовою на побутовому і навіть на професійному рівнях.

Рисунок 1.1 — Опитування, проведене Київським міжнародним інститутом соціології

Дивлячись в майбутнє, створення веб-системи для спрощення та покращення навчального процесу вивчення англійської мови є важливим при подальшому пошуку високооплачуваної роботи. Такими можуть бути:

- керівник компанії, менеджер проєкту;
- журналіст;
- копірайтер;
- менеджер з продажу;
- маркетолог;

– програміст;

– економіст.

Рівень зарплати прямо пропорційно залежить від рівня володіння англійською. Приклад дослідження від Work.ua наведено на рисунку 1.2.

Рисунок 1.2 — Дослідження Work.ua щодо залежності заробітної плати від рівня англійської мови

Підсумовуючи все вище сказане, англійська мова важлива для вивчення з початкової школи. Вона впливає на подальший кар'єрний ріст та можливості які відкриватимуться по всьому світу.

1.3 Аналіз наявних рішень

Під час проектування технічного завдання слід проаналізувати існуючі веб-системи для навчання англійської мови. Дуже важливо виділити наявний

функціонал та відсутній, щоб пізніше мати можливість оптимально спроектувати технічне завдання.

Першим рішенням проаналізуймо Duolingo. Це електронна платформа вивчення мови, що має платний та безплатний види акаунтів. Дуже відомий на весь світ своїм стилем та зеленою совою, яка виступає широковідомим символом, через що і популярність велика.

Перевагами є те, що більшість матеріалу платформи в безплатному доступі, проте. Платна версія пропонує більший функціонал з вивчення англійської мови. До прикладу, можливість відпрацьовувати питання або рівні, в яких допустилася помилка при проходженні. Основна фішка в дизайні платформи це персонажі, які протягом проходження рівнів запрограмовані відтворювати різний текст для симуляції реальної розмови, продемонстровано на рисунку 1.3.

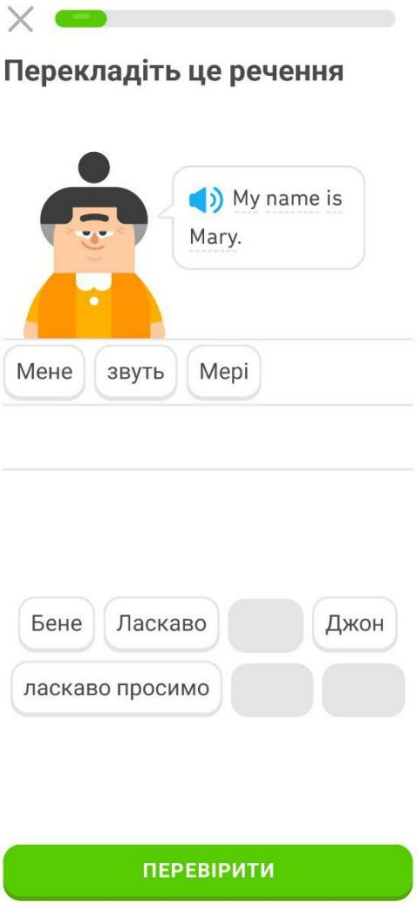

Рисунок 1.3 — Демонстрація проходження рівня в Duolingo

Дизайн та анімація сайту спроектований в ігровому стилі для привертання уваги користувача, що є дуже хорошим стимулятором до навчання дітям.

Наявна система ліг, яка дозволяє змагатись з іншими шляхом накопичення очок, які зачисляються після проходження рівнів. Чим більше очків – тим більший шанс перейти в наступну лігу. Варто відзначити широкий вибір мов для вивчення.Недоліками платформи виступає неможливість на момент аналізу вибір до вивчення інших іноземних мов крім англійської, якщо користувач хоче використовувати інтерфейс та пояснення виключно українською мовою. Це дуже ускладнює процес навчання.

Наступна проаналізована платформа — Kahoot!. Це навчальна платформа в ігровому стилі, яка широко використовується для швидких та цікавих тестувань в навчальних закладах по всьому світу. Переваг тут чимало, одна з таких — наявність каталогу ігор-вікторин, кожна з яких містить питання з декількома варіантами відповіді. Платформа абсолютно безплатна та не має функціоналу, який розблоковується тільки після придбання платної підписки. Перевагою серед своїх конкурентів, платформа отримала через швидкість проведення тестування.

Це відбувається так: користувач, в пошуковій системі пише Kahoot!, заходить на сайт і перше що він бачить це поле, де потрібно ввести код, щоб потрапити на тестування. Це продемонстровано на рисунку 1.4.

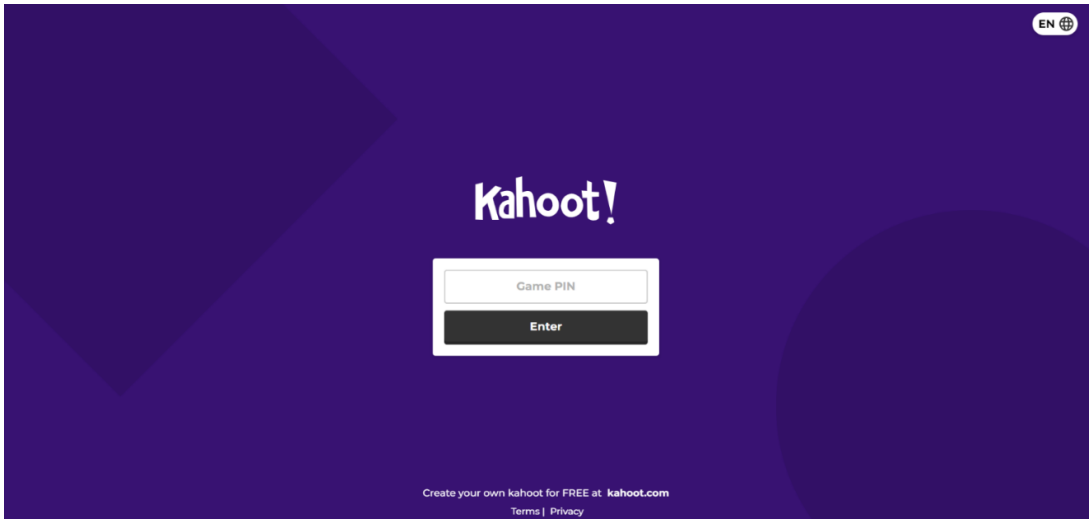

Рисунок 1.4 — Вигляд сторінки входу до тестування на платформі Kahoot!

У випадку, коли опитування проводиться серед дітей або незареєстрованих користувачів — це зручно, не потрібно реєструватись для отримання доступу. Потрібно лише ввести код та вписати ім’я для аутентифікації гравця.

Недоліком є обмежений функціонал під час створення тестувань, можливо створити тест тільки з вибором однієї правильної відповіді (рисунок 1.5).

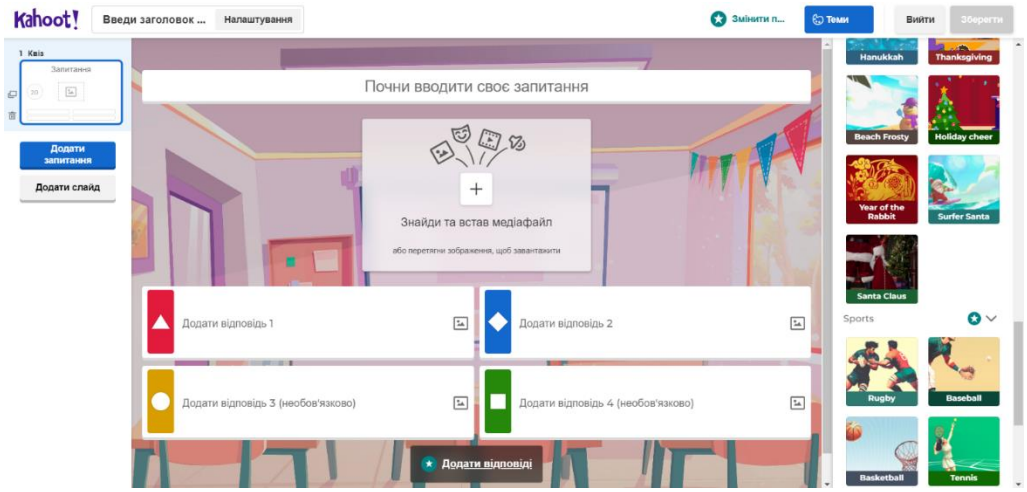

Рисунок 1.5 — Інтерфейс створення тестування на платформі Kahoot!

Платформу не можна назвати націленою на вивчення саме іноземних мов, вона спеціалізується на полегшені проведення опитувань в класі та робить дуже зручною систему під’єднань до тесту на аутентифікації гравців.

Третьою проаналізованою платформою буде Quizlet. Це сервіс, який як і попередній, спеціалізується на покращені та спрощені проведення тестувань. Сервіс дозволяє легко запам’ятати будь-яку інформацію.

Перевагою перед попередніми веб-системами є ширший функціонал при створенні чи проходженні тестувань. Доступне вивчення використовуючи флеш-картки, співвідношення, з вибором однієї правильної відповіді з текстовим чи аудіо умовою, з вибором декількох правильних відповідей та вибір істини чи хибної відповіді.

Це все можна отримати оплативши платну підписку на сайті. Після чого в будь-який момент можна віднайти потрібне тестування та пройти його ще раз. Великим плюсом є можливість створення опитування на основі флеш-карток. Це працює наступним чином: користувач створює або знаходить потрібні картки, вибирає функцію “Створити тест”, та на основі кількості інформації формує на вибір ту кількість на форму питань, яку захоче. Це може бути суцільний вибір однієї правильної відповіді чи різноманітне тестування. Це продемонстровано на рисунку 1.6.

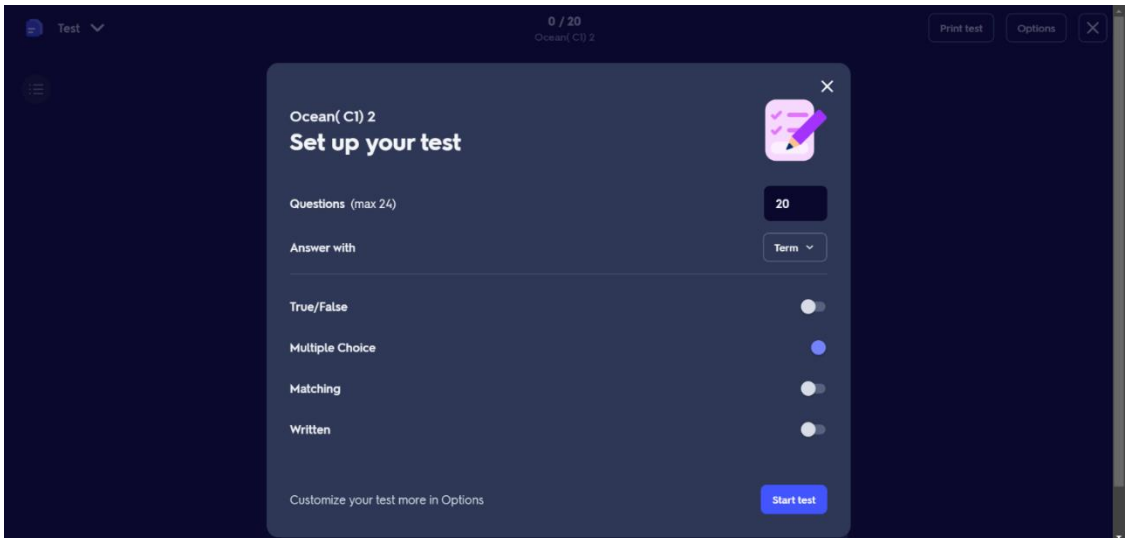

Рисунок 1.6 — Процес створення користувацького тесту на платформі Quizlet

Як альтернатива простим паперовим опитуванням цей сайт є прекрасною варіантом для внесення різноманіття в навчальний процес, що особливо важливо при навчанні з учнями початкових класів.

Недоліками є доволі не дешева ціна за платну підписку на платформі. Також варто звернути увагу на те, що сервіс не спрямований на вивченні саме англійської мови, це може бути будь-що інше. На сайті немає функції разового проходження чи аутентифікації користувача, що ускладнює процес встановлення правдивості тестування.

1.4 Аналіз вимог та постановка завдання

Обміркувавши всі переваги та недоліки платформ які допомагають вивчати англійську мову, можна виділити основні вимоги до веб-системи.

Перш за все, дизайн веб-системи повинен бути в ігровому стилі, що дозволить привернути увагу школярів та зробити процес навчання цікавим. Лиш тільки так можна досягнути більших результатів. Кольори та загальне оформлення не повинно викликати відрази або бажання якнайшвидше залишити платформу.

Для вчителів повинна бути можливість простого та швидкого створення тестувань з різними типами питань. Це може бути як з вибором однієї або декількох правильних відповідей так і встановлення співвідношення. Для учнів ж можливість пройти тестування. Після цього система повинна видати персональний результат. На сторінці особистого профілю реалізувати можливість перегляду всіх пройдених раніше тестувань та їх результати. Таким чином вся необхідна інформація буде в зручному місці та доступ до якої немає часових обмежень.

На окремій сторінці створити список з всіх існуючих тестувань, які були створені раніше. Це зроблено для розширення можливостей навчання в індивідуальних цілях.

Реєстрація та вхід користувача в веб-системі повинен реалізовуватися зручно, щоб батькам, вчителям чи дітям не було важко пройти процес реєстрації. Обов’язково повинне використовуватись шифрування паролів задля зберігання конфіденційної інформації кожного користувача.

Навігація по сайту не повинна складатись з багатьох елементів. Це може погано вплинути на сприйняття школярів. Все повинно бути максимально просто та доступно.

На рисунку 1.7 продемонстровано варіанти використання веб-системи.

Кожен користувач має свої особливі можливості. Весь функціонал який доступний учневі:

– проходження тестування;
– отримання результату;
– реєстрація/вхід;
– перегляд особистого профілю;
– перегляд всіх тестувань.

Функціонал вчителя дозволяє створювати тестування, що є основним для даного користувача. Далі перераховано доступний функціонал:

– проходження тестування;
– отримання результату;
– реєстрація/вхід;
– перегляд особистого профілю;
– перегляд всіх тестувань;
– створення тестувань.

Рисунок 1.7 — Варіанти використання веб-системи

Адаптація під мобільні додатки теж важливий крок який не можливо не оминути. Для ефективного використання веб-системи на будь-яких пристроях це обов’язково для реалізації.

Підсумовуючи все вище сказане, веб-система повинна мати функціонал створення, проходження та отримання результатів тестувань, мати доступ до особистого профілю.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Формалізація вимог

Формалізація функціональних вимог базується на технічних потребах кінцевого продукту, таких як, можливість реєстрації/входу в систему. Доступ до можливостей системи та інформації надаватиметься залежно від типу користувача. Інформації необхідної для реєстрації та входу потрібно використовувати мінімально, зважаючи на те, що система розроблятиметься в більшості для школярів молодших класів, хоча під чітким керуванням вчителя та допомоги зі сторони батьків це не спричинятиме великих проблем.

На рисунку 2.1 продемонстровано діаграму послідовності реєстрації/входу користувача.

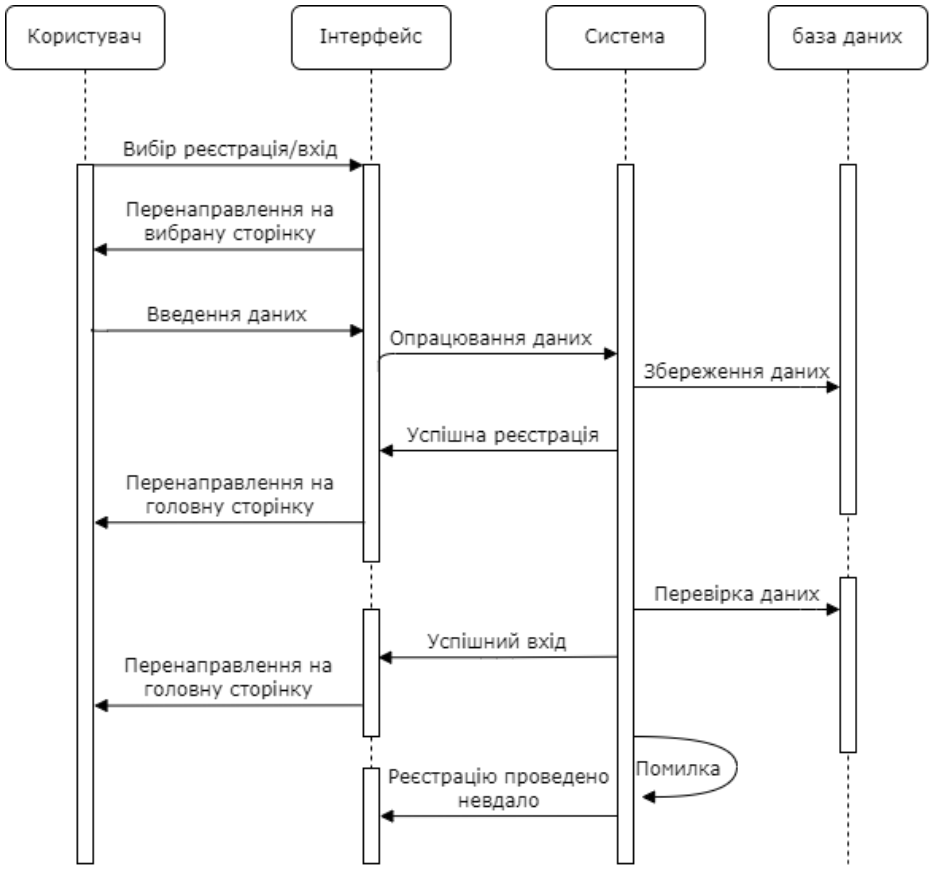

Рисунок 2.1 — Діаграма послідовності реєстрації/входу в систему

Доступ до перегляду особистого профілю. На даній сторінці розташовуватиметься особиста інформація, з метою надання користувачу можливості ідентифікувати акаунт або можливі помилки введених даних. Також розміщення інформації про пройдені тестування.

Доступ до перегляду навчального матеріалу. Можливість переглядати теоретичну інформацію, створену вчителями, з метою пояснення та вивчення матеріалу. Може проводитись як на уроці, так і самостійно вдома.

Можливість створювати/проходити розділи тестування. Тестування, спрямоване на підтвердження учнем засвоєння інформації. Можливість окремою групою користувачів створювати тестування. Можливість проходження надаватиметься всім групам користувачів.

Доступ до навігаційної панелі системи. Основним способом навігації по системі буде саме навігаційна панель. Метою буде створення мінімальної кількості розділів навігація для полегшення процесу знаходження необхідної інформації;

Нефункціональні вимоги однаково важливі як і функціональні. Попередній аналіз показує необхідність чітко сформувати пункти, такі як, простий та привабливий дизайн системи. Яскрава колірна гама, простий дизайн необхідний для мотивації учнів брати участь у процесі навчанні використовуючи систему для отримання інформації.

Зручна та інтуїтивно зрозуміла структура системи. Очікуючи те, що учні молодших класів ще не мають достатнього досвіду для самостійного пошуку тої чи іншої інформації оптимізація та спрощення структури системи необхідно реалізувати.

Мобільна адаптація. Один з найважливіших пунктів, зважаючи на те, що під час уроків і вдома учні не матимуть можливості придбати або використовувати персональні комп’ютери чи ноутбуки під час користування системою.

Формалізація вимог важливе на етапі проєктування системи. Наступні кроки залежать від правильно попередньо виконаних етапів. Потрібно врахувати всі можливі проблеми, потреби які виникатимуть під час реалізації або кінцевого користування продуктом.

2.2 Проєктування структури

В даній системі важливо виділити лише найнеобхідніші елементи та розташувати їх в інтуїтивно зрозумілих модулях. Варто зауважити, що всі сторінки крім реєстрації та входу матимуть доступ до компонентів навігації системою, тому далі це не прописано. Отже, структура спроєктована наступним чином:

Сторінка реєстрації. Перше що побачить новий користувач — саме цю сторінку. Система не даватиме доступ до матеріалів без акаунту. Компоненти які будуть використовуватимуться:

- поле для вводу даних;
- кнопка підтвердження реєстрації;
- кнопка переходу на сторінку для входу.

Сторінка входу. Якщо користувач вже має створений акаунт, за допомогою даної сторінки можна увійти в систему, після чого буде відображена сторінка з навчальними матеріалами. На цій сторінці розміщуватимуться:

- поле для вводу даних;
- кнопка підтвердження входу;
- кнопка переходу на сторінку для реєстрації.

Особистий профіль. Відображатиме інформацію про користувача. Також, в іншій частині сторінки розміщуватиметься історія пройдених тестувань, для зручності відстеження. Компоненти будуть наступні:

- блок з особистою інформацією;
- блок з пройденими тестуваннями;
- кнопка виходу з профілю.

Навчальні матеріали. Основна сторінка, на якій буде відображено всі доступні теми для вивчення. Припускається, що керувати процесом навчання буде вчитель, за власним розсудом, тому система не надаватиме можливості створити стрічку з тем чи чогось подібного. Відображення буде в вигляду списку з переліком всіх тем на вибір. Компонентами на даній сторінці буде:

- список тем;
- кнопка переходу сторінку відповідної теми.

Сторінка вибраної теми. Сторінка, яка відображатиме всю додану раніше навчальну інформацію для вивчення. Також, окремо розташовуватиметься кнопка для проходження тестування, лише за умови, що користувач який створював тему додав та відредагував тестування. Компонентами саме тут буде лише кнопка повернення на попередню сторінку, тобто, навчальні матеріали та кнопка яка переадресує користувача на іншу сторінку для початку проходження тестування.

Сторінка проходження тестування вибраної теми. Сторінка, яка відкриватиметься для проходження користувачам. Як було сказано в попередньому пункті — доступ можна отримати натиснувши відповідну кнопку на сторінці теми. Складатися сторінка буде з:

- блок з текстовою інформацією питання;
- кнопки переходу до наступного/попереднього запитання;
- поле для введення відповіді;
- кнопка завершення тестування та отримання результату.

Саме таким чином виглядає структура даної веб-системи. Візуально діяльність можна переглянути на рисунку 2.2.

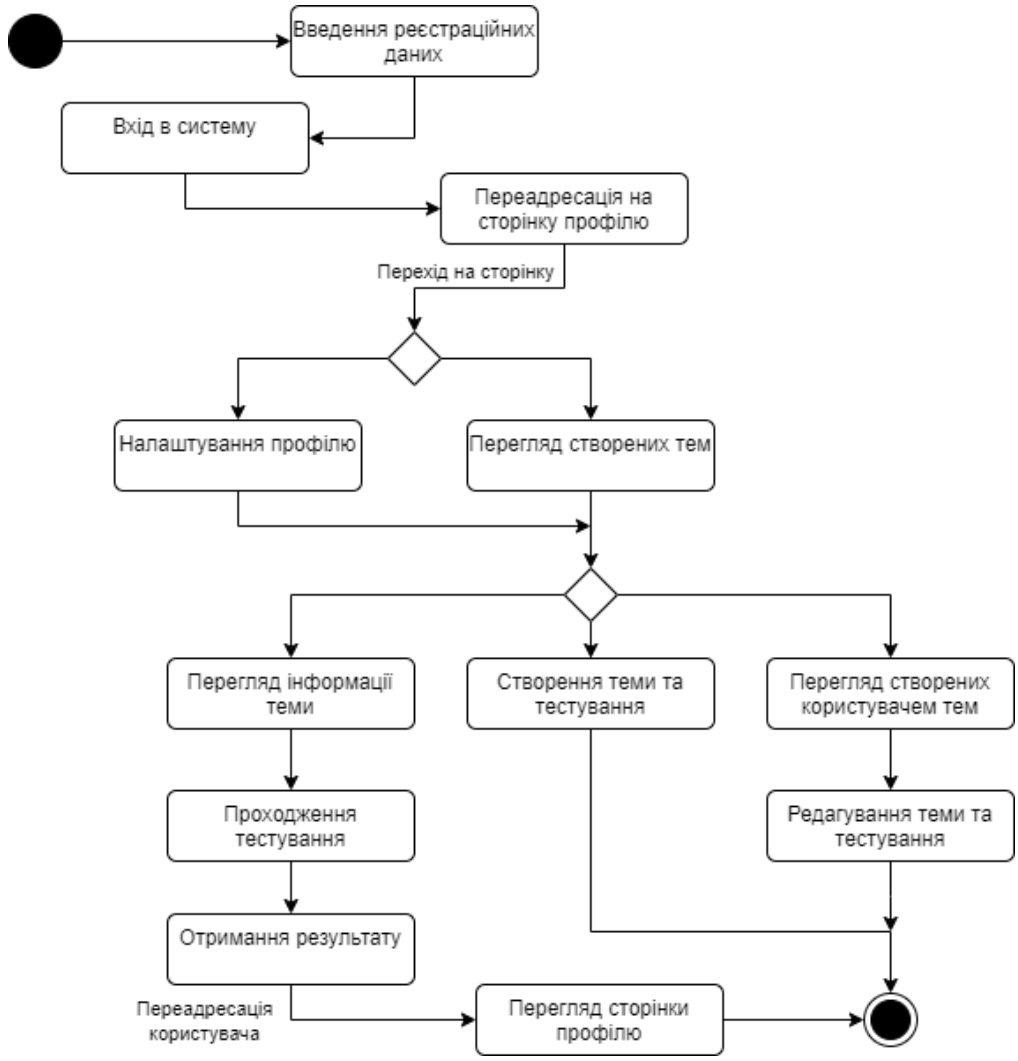

Рисунок 2.2 — Діаграма діяльності веб-системи

Дуже чітко та зрозуміло продемонстровано діяльність користувачів, компонентів та зв’язки між сторінками та процесами. Таким чином, детальний опис структури веб-системи та її компонентів забезпечує чітке уявлення про дії користувача та взаємозв’язки між різними функціональними сторінками та компонентами, що сприяє ефективному впровадженню та розвитку системи.

2.3 Аналіз технологій реалізацій

Технологій для вирішення схожих проблем існує безліч, але деякі з них виділяються серед всіх. Деякі своєю простотою в використанні, деякі в легкості вивчення, якісь великою кількістю додаткових бібліотек.

Проте, не варто використовувати будь що, окремі технології потребують спеціальних знань та можуть застосовуватись лише для обмеженого списку завдань.

Для даної системи потрібно простота та ефективність. Ще одним показником при виборі засобу реалізації інтерфейсу буде можливість використовувати компоненти для мобільної адаптації.

HTML — це мова, яка використовується для створення веб-сторінок: вона керує їх структурою та вмістом. Гіпертекст відноситься до текстової інформації, пов’язаної з іншим текстом за допомогою посилань, сплетення мережі взаємопов’язаних сторінок.

Мова розмітки допомагає браузерам ідентифікувати текстову інформацію та представляти її читачам у зручний спосіб. Існують інші мови гіпертекстової розмітки, але переважна більшість сторінок в Інтернеті написані на HTML.

CSS (скорочення від Cascading Style Sheets) — це спеціальна мова (мова стилів), яка використовується для опису зовнішнього вигляду (як і де з’являються елементи веб-сторінки) документа, написаного мовою розмітки даних. CSS найчастіше використовується для документів, розмічених у HTML, XHTML та XML.

Однією з головних переваг використання CSS є можливість відокремити вміст сторінки від її дизайну [1].

Bootstrap — це безкоштовний і відкритий фреймворк HTML, CSS і JS для швидкого створення макетів і адаптивного дизайну веб-сайтів і WEB-додатків. Bootstrap дуже популярний серед розробників. Це дозволяє створювати веб-сайти у багато разів швидше, ніж використання чистого CSS і JavaScript. добре розроблена зовнішня структура, яку можна легко налаштувати за допомогою збірки Sass. Навіть новачки (без великих знань і достатньої практики) можуть швидко створити макети досить хорошої якості.

Фреймворк Bootstrap представляє собою набір CSS та JavaScript файлів. Щоб його використовувати ці файли, необхідно просто підключити їх до сторінки або включити аналогічний функціонал з CDN (Content Delivery Network).

Перевагами Bootstrap є швидкість розробки та гнучкість. Фреймворк надає багато можливостей для налаштування дизайну і поведінки елементів веб-сторінки та дозволяє швидко створювати стилізовані і адаптивні веб-сторінки за рахунок готових компонентів і шаблонів. Також, Bootstrap має відкритий вихідний код, який доступний на Github. Він має ліцензію MIT. Це означає, що його можна безкоштовно використовувати як у відкритих, так і комерційних застосунках.

Недоліками ж будуть: залежність від фреймворка та збільшення розміру файлів. Використання Bootstrap може призвести до того, що веб-сайт буде схожий на інші сайти, що також використовують цей фреймворк. Також, включення всіх необхідних файлів Bootstrap може призвести до збільшення обсягу сторінки і збільшення часу завантаження [3].

Node.js — це однопоточне кросплатформове середовище виконання з відкритим вихідним кодом і бібліотека, яка використовується для запуску веб-додатків, написаних на JavaScript, поза браузером клієнта.

Простіше кажучи, Node.js — це програмне середовище, яке дозволяє запускати програми, написані мовою Javascript, поза браузером. Історично програми, написані на Javascript, на відміну від інших мов програмування, можна було запустити лише у браузерах, які мали спеціальний вбудований движок виконання коду цієї мови. Поза браузером Javascript, можна сказати, не працював.

Переваги Node.js серед інших технологій є масштабованість та використання JavaScript. Велика кількість розробників вже володіє знаннями JavaScript, що полегшує вивчення Node.js. Також надається можливість створювати швидкі і масштабовані веб-додатки завдяки своїй асинхронності і подієвому принципу [7].

Недоліками в основному є однопотоковість та велике споживання пам'яті. Деякі застосунки на Node.js можуть споживати більше пам'яті через асинхронний стиль програмування, а оскільки він однопотоковий то він може виявитися неефективним для обробки великої кількості одночасних запитів.

2.4 Проєктування інтерфейсу

Зважаючи на досвід проаналізованих раніше наявних рішень та програмних забезпечень які використовувалися чи використовуються зараз в навчальних цілях, мінімізувати кількість кроків для виконання тої чи іншої дії є в пріоритеті.

Почнемо з проєктування інтерфейсу сторінок для входу (рисунок 2.3) та реєстрації (рисунок 2.4). Оскільки користувач не зацікавлений залишатися довго на цьому етапі, розташування функціонала повинне бути разом. Обов’язково на кожній потрібно розмістити кнопки які переадресовують користувача на іншу сторінку, тобто з реєстрації на вхід і навпаки.

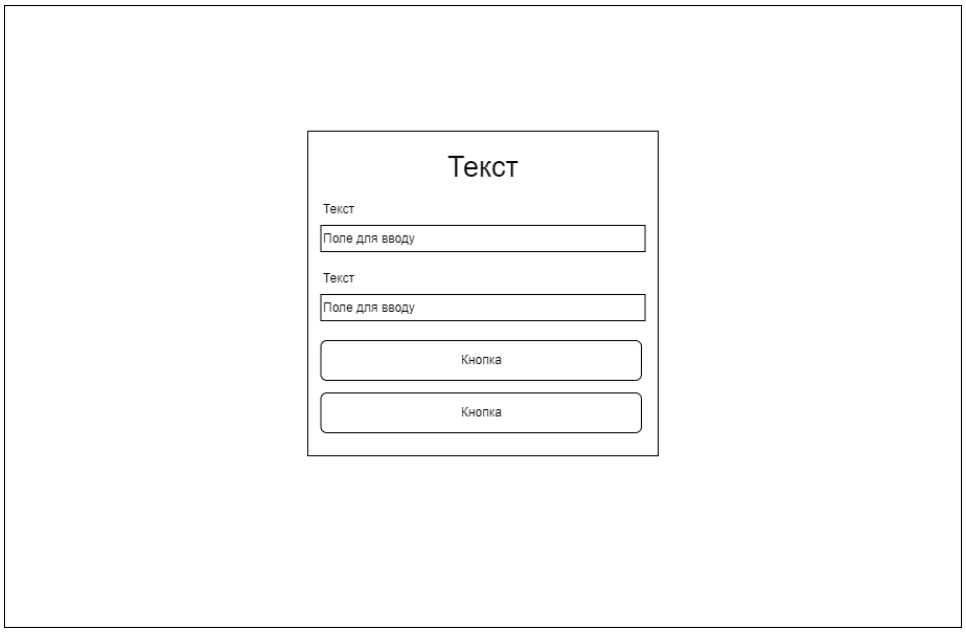

Рисунок 2.3 — Макет сторінки входу

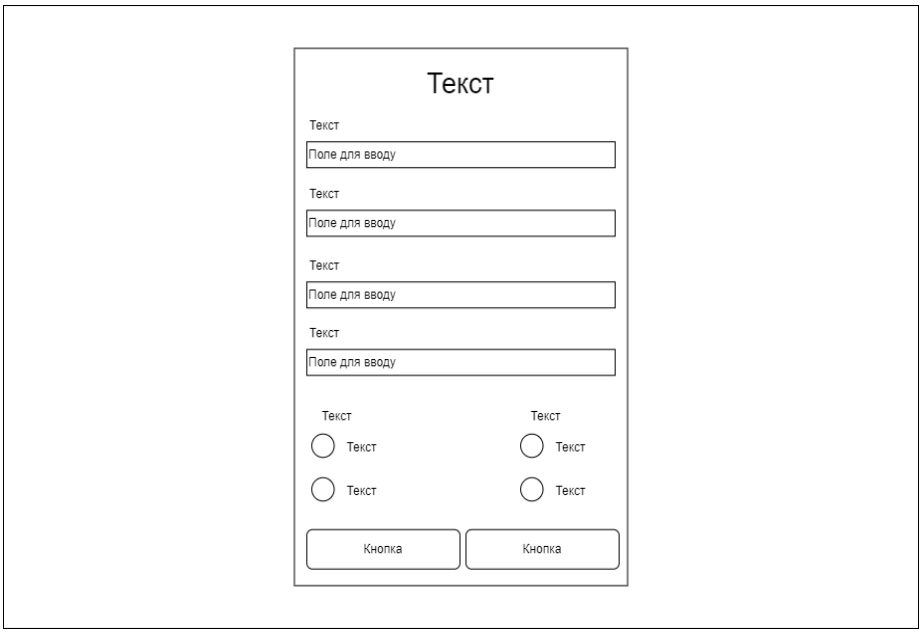

Рисунок 2.4 — Макет сторінки реєстрації

На всіх наступних сторінках буде доступним панель керування для переміщення між сторінками.

Особистий профіль, сторінка міститиме особисту інформацію користувача (ім’я, прізвище, логін), також групу до якої належить користувач, це може бути як вчитель так і учень. Попередньо визначено, що додаткова інформація про пройдені тестування як окремий блок. Виглядатиме це наступним чином (рисунок 2.5).

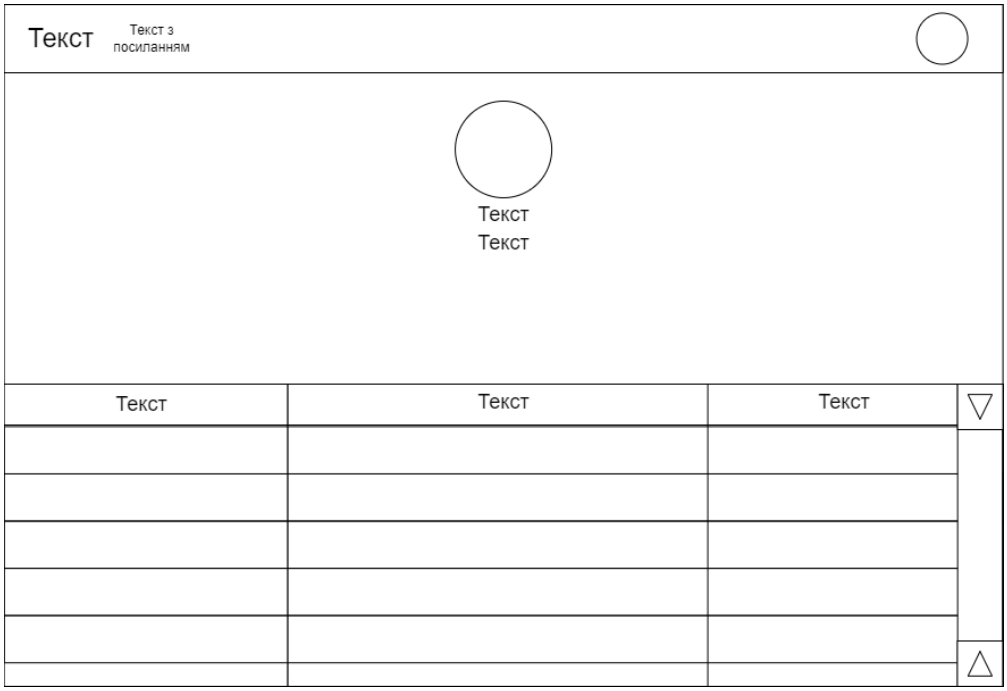

Рисунок 2.5 — Макет сторінки особистого профілю

Сторінка переліку тем повинна містити найнеобхіднішу інформацію. Розміщення елементів, таких як, навігаційна панель та розділи з темами, повинні спрощувати процес користування веб-системою. Макет виглядатиме так, як продемонстровано на рисунку 2.6.

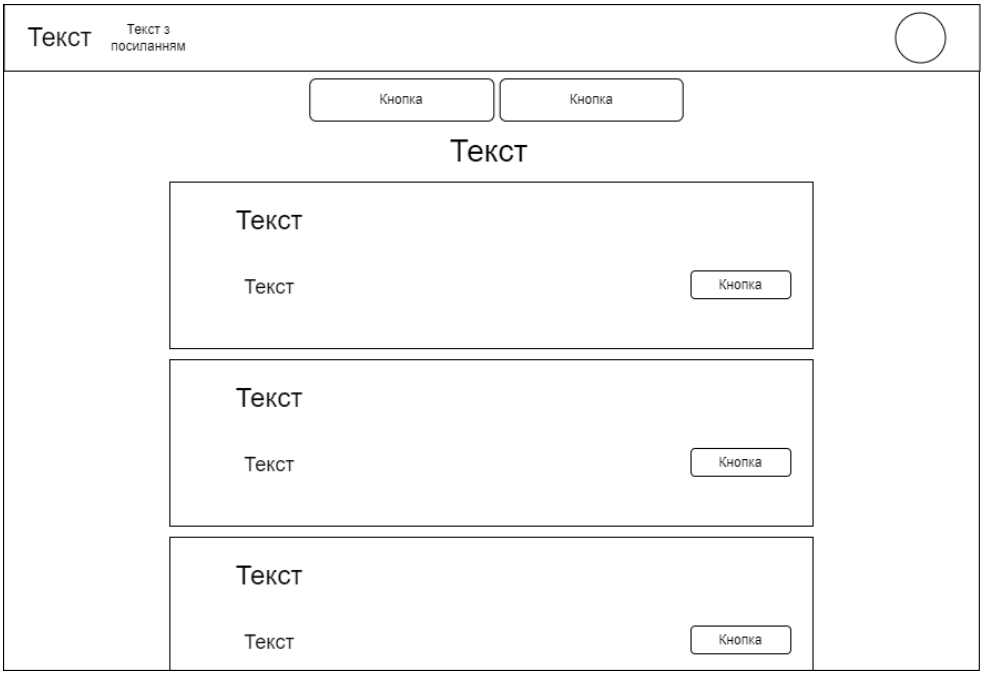

Рисунок 2.6 — Макет сторінки «перелік тем»

Сторінка перегляду теми міститиме інформацію додану іншим користувачем. Кнопка повернення до попередньої сторінки, навігаційна панель відіграватимуть роль інструмента переміщення.

Додаткова кнопка для перенаправлення на іншу сторінку — проходження тестування по вибраній тему, розміщуватиметься на початку. Це зроблено для тих випадків, коли користувач вже пройшов матеріал або бажає почати тестування одразу.

Макет сторінки виглядатиме наступним чином (рисунок 2.7).

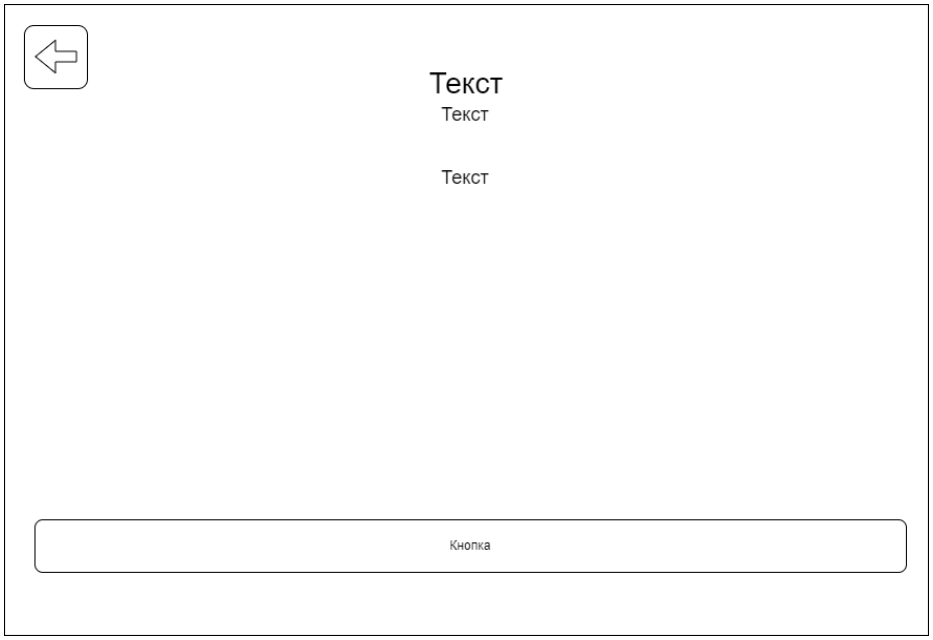

Рисунок 2.7 — Макет сторінки теми

Сторінка проходження тестів міститиме текст питання та функціональні кнопки для взаємодії. Також поле для вводу відповіді на відповідне питання. В будь який момент користувач матиме можливість натиснути кнопку для завершення тестування. Макет виглядатиме наступним чином (рисунок 2.8).

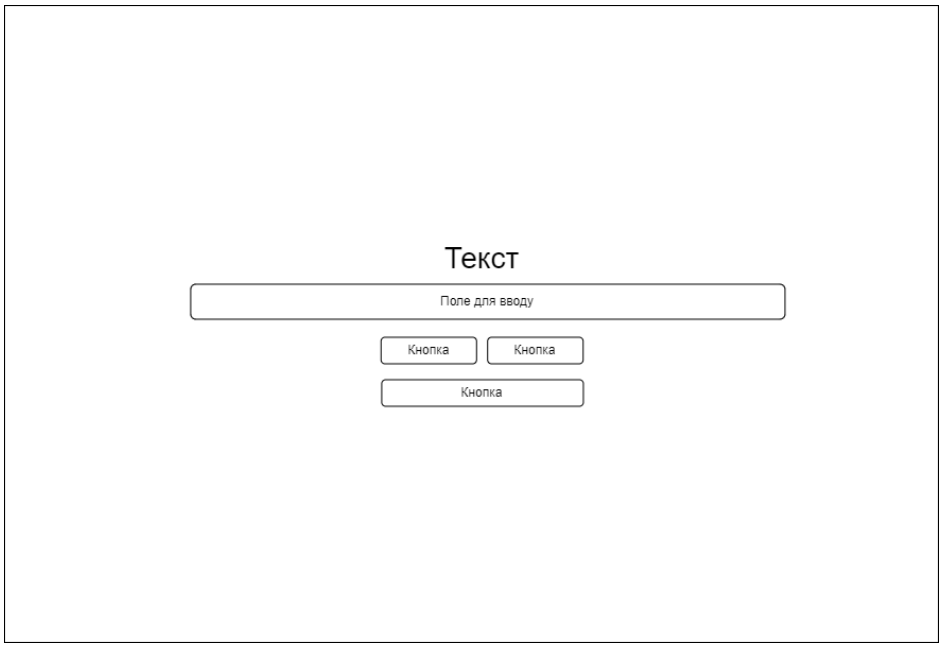

Рисунок 2.8 — Макет сторінки проходження тестування

Сторінка створення тестувань та теми міститиме функціонал з використанням введення користувацького тексту. Елементи розміщуватимуться максимально компактно та виглядатимуть наступним чином (рисунок 2.9).

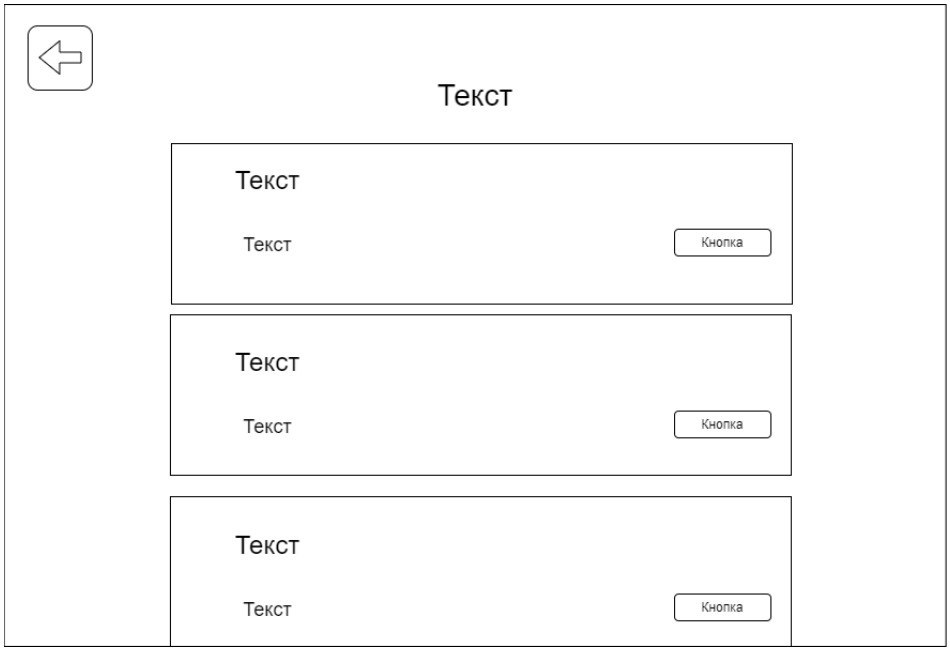

Рисунок 2.9 — Сторінка тем, створених користувачем

Для зручності, варто спроєктувати сторінку, на якій відображатимуться всі теми, які створив певний користувач. Макет дещо схожий з переліком тем, проте матиме в переліку обмежену інформацію (рисунок 2.10).

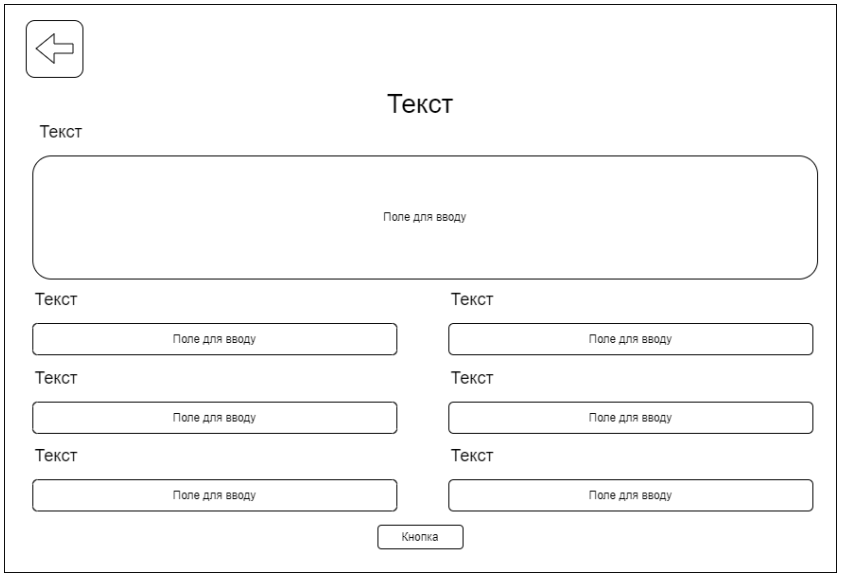

Рисунок 2.10 — Макет сторінки створення теми та тестування

Для подальшої взаємодії з створеними темами, користувачу буде запропоновано перейти на наступну сторінку (рисунок 2.11), та в ручну відредагувати тему.

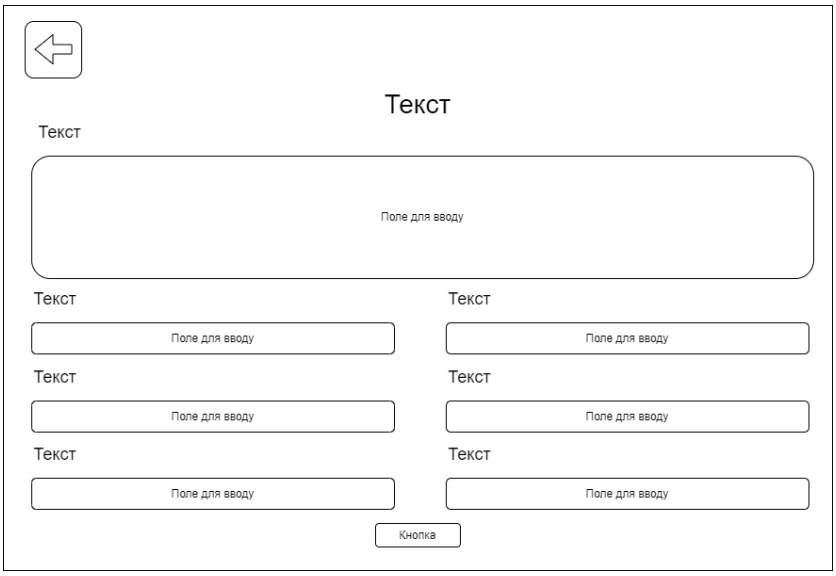

Рисунок 2.11 — Сторінка редагування теми

Для того, щоб користувач мав змогу налаштувати деякі дані зв’язані з його особистим профілем, буде реалізована сторінка налаштувань особистого профілю (рисунок 2.12). На якій буде розміщено ряд функцій для змінення тої чи іншої характеристики користувача.

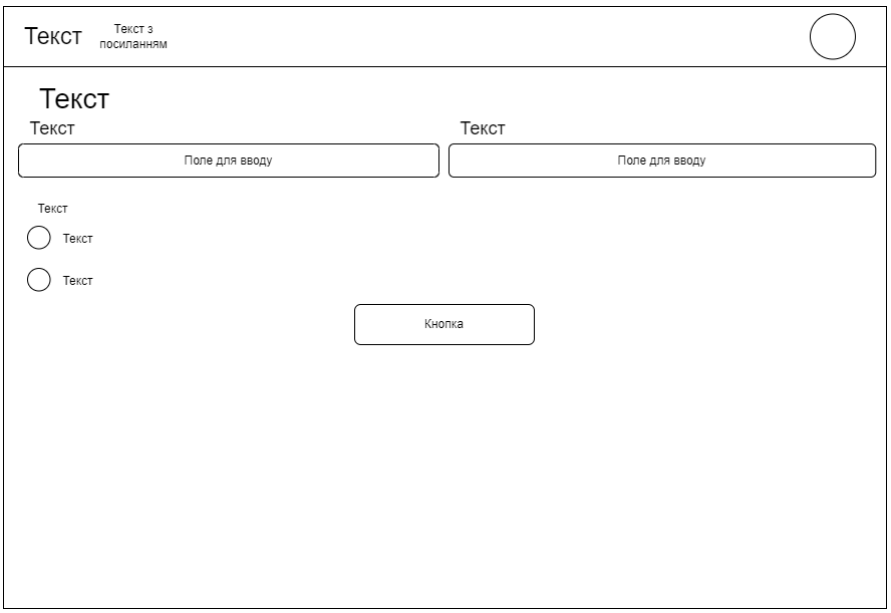

Рисунок 2.12 — Сторінка налаштувань особистого профілю

Після проведеного дослідження та проєктування, створено макети сторінок для веб-системи, які в подальшому будуть використовуватися в реалізації програмним кодом.

2.5 Проєктування бази даних

При розробці будь-якого інформаційного системи ключовим етапом є проєктування бази даних, яке визначає основну структуру, організацію та зв'язки між даними. Ціллю є забезпечення бази даних, яка буде надійним фундаментом для роботи та реалізації всієї системи забезпечуючи її стабільність, продуктивність та гнучкість.

Цим етапом передбачено створення таблиць та полів, які взаємодіятимуть між собою один одного та мінімізують кількість потрібних комірок з даними, тобто ті, які відповідатимуть всім вимогам. Далі продемонстровано який зміст міститиме база даних в таблиця 2.1, таблиця 2.2, таблиця 2.3 та таблиця 2.4.

Таблиця 2.1 — Таблиця з даними користувача

user		
Назва поля	Тип даних	Ключове поле
user_id	Число	Так
user_name	Короткий текст	Ні
user_surname	Короткий текст	Ні
user_gender	Короткий текст	Ні
user_login	Короткий текст	Ні
user_password	Короткий текст	Ні
user_permition_level	Число	Ні

Таблиця 2.2 — Таблиця з даними про спроби проходжень тестувань

attempts_to_pass_topic_test		
Назва поля	Тип даних	Ключове поле
attempt_id	Число	Так
user_id	Число	Ні
topic_test_id	Число	Ні
attempt_result	Число	Ні

Таблиця 2.3 — Таблиця з даними про тестування тем

topic_test_information		
Назва поля	Тип даних	Ключове поле
topic_test_id	Число	Так
topic_question_one	Довгий текст	Ні
topic_answer_one	Довгий текст	Ні
topic_question_two	Довгий текст	Ні
topic_answer_two	Довгий текст	Ні
topic_question_three	Довгий текст	Ні
topic_answer_three	Довгий текст	Ні
topic_question_four	Довгий текст	Ні
topic_answer_four	Довгий текст	Ні
topic_question_five	Довгий текст	Ні
topic_answer_five	Довгий текст	Ні
topic_question_six	Довгий текст	Ні
topic_answer_six	Довгий текст	Ні

Таблиця 2.4 — Таблиця з даними про тему

topic_information		
Назва поля	Тип даних	Ключове поле
topic_id	Число	Так
topic_name	Короткий текст	Ні
topic_description	Текст	Ні
topic_creator_information	Число	Ні
topic_main_information	Довгий текст	Ні
topic_test_id	Число	Ні

Наступним рисунком (рисунок 2.17) продемонстровано як саме взаємопов’язані, раніше показані, таблиці. Вимоги та функціонал не вимагають складних рішень для реалізації саме даної веб-системи.

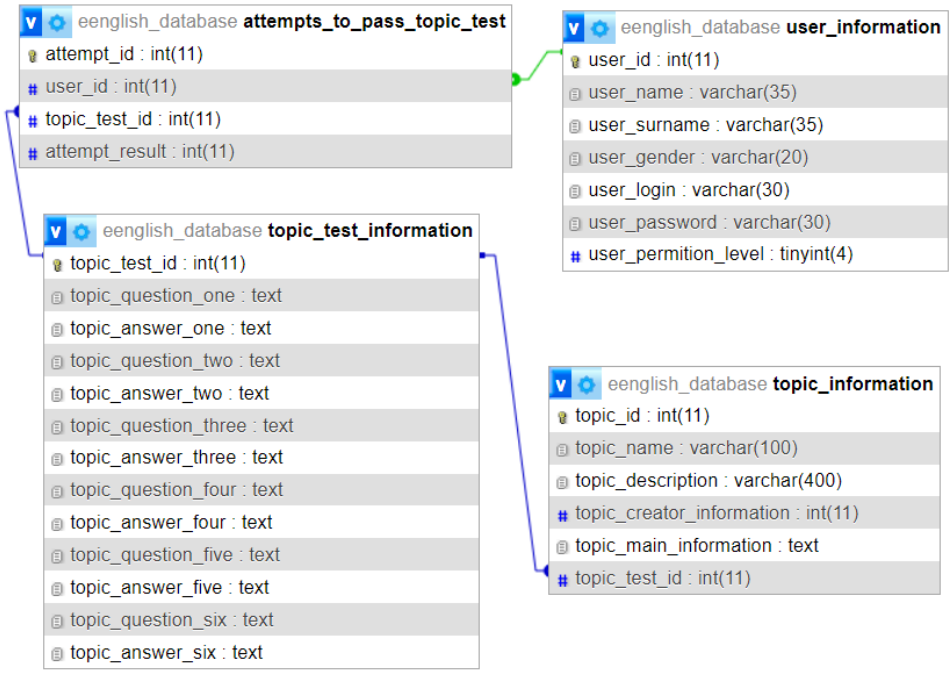

Рисунок 2.17 — Фізична модель бази даних

У висновку цього підрозділу слід зазначити, що проєктування бази даних відіграє ключову роль у забезпеченні ефективності, надійності і гнучкості інформаційних систем. Всі основні кроки проєктування були успішно описані та впроваджені.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

Розділ реалізації та тестування системи є ключовим. В ньому відображено весь процес використання новітніх технологій реалізації функціоналу та полегшення взаємодії користувачем з системою.

3.1 Реалізація інтерфейсу

Інтерфейс для системи, яка прогнозована використовуватись великим колом користувачів, опираючись на можливий брак досвіду роботи з схожими платформами, повинна бути інтуїтивно зрозумілою.

Рішень для побудови інтерфейсу існує безліч, як і варіантів їх використання. Одним із найпростіших для використання фреймворків є BootStrap. Перевагою буде вбудована адаптація інтерфейсу та дизайну під мобільні пристрої. В реалізації даної системи це важливо через те, що переважною більшістю користувачів будуть саме учні, які користуватимуться системою під час уроків.

Передбачаючи, що уроки будуть проводитись не в приміщеннях оснащеними персональними комп’ютерами дуже важливим буде саме адаптування.

Перед цим, використовуючи мову розмітки HTML, потрібно реалізувати, так званий «каркас» системи. Сторінки не будуть мати належного привабливого вигляду, вони міститимуть лише функціональні поля, з якими можна взаємодіяти, приблизний порядок розташування об’єктів такі як фотографія, кнопка, таблиці, поля для вводу та текстові поля.

Для цього, існують групи тегів <form>, які призначені для розміщення функціональних об’єктів, як поле, кнопка і тому подібне. Основними тегами для реалізації будуть:

– Input. Це один із найпоширеніших елементів форм, який застосовується для різних типів полів, залежно від їх призначення.

– Label. Використовується для визначення підпису для поля форми. Це допомагає користувачам легко ідентифікувати елементи введення. Елемент label зазвичай асоціюється з елементом input.

– Select. Створює випадаючий список, дозволяючи користувачам вибирати один або кілька варіантів з переліку.

– Text area. Забезпечує багаторядкове текстове поле, яке часто використовується для збору коментарів або відгуків. Розмір цього поля можна налаштувати за допомогою відповідних атрибутів.

– Button. Створює натискну кнопку, яку можна використовувати для різних дій у формі.

– Dropdown menus. Використовуються для створення списків з варіантами, що випадають, забезпечуючи можливість вибору з декількох опцій.

– Radio buttons. Використовуються для вибору одного варіанту з групи можливих, часто застосовуються для вказання статі або інших одноразових виборів.

Для початку, потрібно використати все вище перераховане та створити сторінку для реєстрації користувача. Розміщуватимуться на сторінці такі елементи:

– Input, для вводу інформації про користувача та можливості отримати серверній частині достатньої кількості інформації для її опрацювання та додавання нового користувача.

– Radio button використовуватиметься на даній сторінці лише для того, щоб користувач мав змогу вказати стать.

– Текстові поля, для надання додаткової інформації, що допоможе користувачеві зорієнтуватись в діях.

– Button, для двох завдань, підтвердження реєстрації користувача та переходу на сторінку входу, за умови наявності в користувача профілю.

Без використання CSS та фреймворку Bootstrap, розміщення елементів дещо відрізнятиметься, тому сторінка реєстрації виглядатиме як на рисунку 3.1.

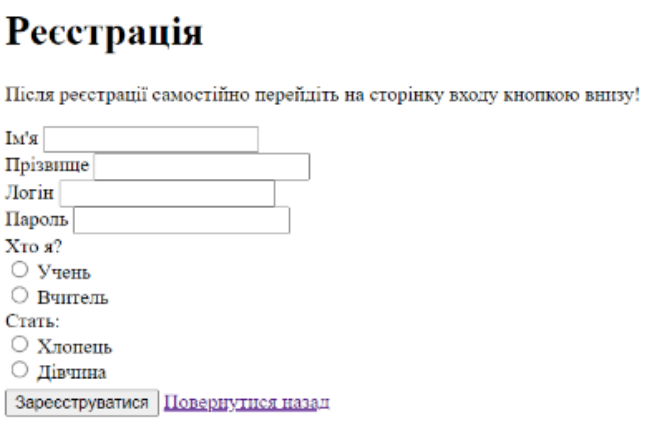

Рисунок 3.1 — Вигляд сторінки «реєстрації» з використанням форм

Далі, продемонстровано код даної сторінки.

Надпис та підказка для користувача:

```
<h1 class="large-h1">Реєстрація</h1>

            <p>Після реєстрації самостійно перейдіть на
сторінку входу кнопкою внизу!</p>
```

Поля для вводу даних користувача:

```
<div class="mb-3">

                <label for="" class="form-label-
custom">Ім'я</label>

                <input type="text" class="form-control"
name="accName" id="accName" aria-describedby="helpId"
placeholder="" />

            </div>

            <div class="mb-3">

                <label for="" class="form-label-
custom">Прізвище</label>

                <input type="text" class="form-control"
name="accSurname" id="accSurname" aria-describedby="helpId"
```

```
placeholder="" />
            </div>
            <div class="mb-3">
                <label for="" class="form-label-
custom">Логін</label>
                <input type="text" class="form-control"
name="accLogin" id="accLogin" aria-describedby="helpId"
placeholder="" />
            </div>
            <div class="mb-3">
                <label for="" class="form-label-
custom">Пароль</label>
                <input type="password" class="form-control"
name="accPassword" id="accPassword" aria-describedby="helpId"
placeholder="" />
            </div>
```

Вибір типу користувача. В подальшому це впливатиме на можливість створювати та редагувати теми та тести:

```
<div class="form-check">
                            Хто я?<br>
                            <input class="form-check-input"
type="radio" name="flexRadioDefault" id="flexRadioDefault1"
value="Учень">
                            <label class="form-check-label"
for="flexRadioDefault1">
                              Учень
                            </label>
                          </div>
                          <div class="form-check">
                            <input class="form-check-input"
type="radio" name="flexRadioDefault" id="flexRadioDefault2"
value="Вчитель">
                            <label class="form-check-label"
for="flexRadioDefault2">
```

```
                                Вчитель


 </label>
                              </div>
```

Вибір статі користувача:

```
                            <div class="form-check">
                                Стать:<br>
                                <input class="form-check-input"
type="radio" name="flexRadioDefault1" id="flexRadioDefault11"
value="Хлопець">
                                <label class="form-check-label"
for="flexRadioDefault1">
                                  Хлопець
                                </label>
                              </div>
                              <div class="form-check">
                                <input class="form-check-input"
type="radio" name="flexRadioDefault1" id="flexRadioDefault22"
value="Дівчина">
                                <label class="form-check-label"
for="flexRadioDefault2">
                                  Дівчина
                                </label>
                              </div>
```

Кнопки:

```
                           <button style="margin-top: 5px;"
type="submit" class="btn btn-primary"
onclick="registerUser()">
                                Зареєструватися
                           </button>
                        <a style="margin-top: 5px;"
href="./login_page.html" class="button btn btn-primary">
```

```
                Повернутися назад
            </a>
```

Деяка кнопка матиме функціонал переходу на сторінку входу, як було вказано раніше, тому потрібно створити цю саму сторінку для входу в існуючий профіль. На даній сторінці міститиметься всього лиш два поля для вводу даних, текстові поля та кнопки. Виглядатиме сторінка, як на рисунку 3.2.

Рисунок 3.2 — Вигляд сторінки «входу» з використанням форм

Код текстового поля:

```
<h1 class="large-h1">Вхід</h1>
```

Код полів для вводу даних:

```
        <div class="mb-3">
            <label for="" class="form-label-
custom">Логін</label>
            <input type="text" class="form-control" name=""
id="input_userLogin" aria-describedby="login_input"
placeholder=""/>
        </div>



        <div class="mb-3">
```

```
            <label for="" class="form-label-
custom">Пароль</label>
            <input type="password" class="form-control"
name="" id="input_userPassword" aria-
describedby="password_input" placeholder=""/>
        </div>
```

Код кнопок:

```
                <button type="submit" class="btn btn-primary"
onclick="send_login_request()">
                    Увійти
                </button>
                <div class="form-button-custom1">
                    <a href="./register_page.html"
class="button btn btn-primary">
                        Зареєструватися
                    </a>
```

Відповідно до підпису на кожній з кнопок, «Увійти» матиме функціонал для підтвердження введених даних та входу в систему, а «Зареєструватися», переадресовуватиме користувача на сторінку «реєстрації».

Наступною сторінкою, після входу користувача, буде «особистий профіль». Згідно макету, розміщуватимуться таблиці, кнопки, текстові поля та елементи, з якими можна взаємодіяти. Виглядатиме сторінка, як на рисунку 3.3.

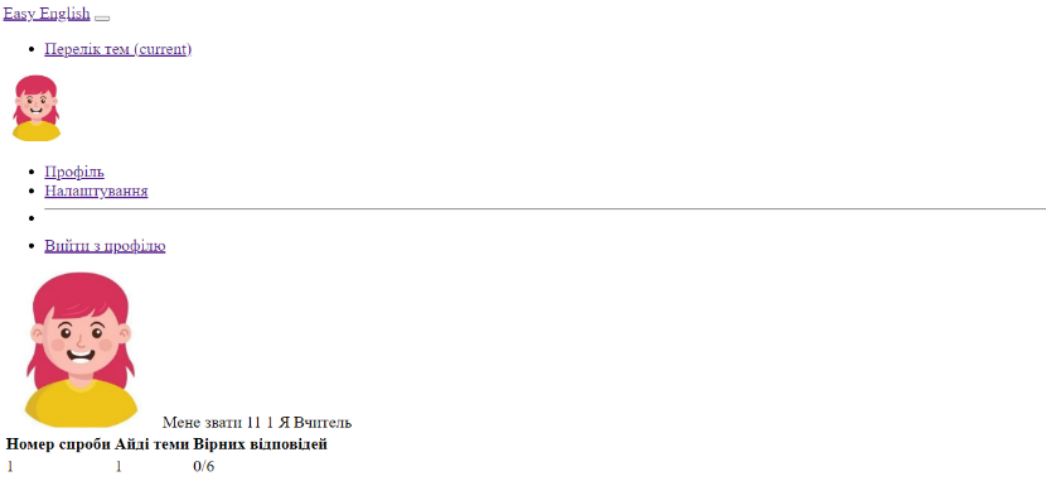

Рисунок 3.3 — Вигляд сторінки профілю з використанням форм

На деяких з наступних сторінок, включаючи дану, буде повторюватись код, що відповідатиме за навігаційну панель. Розміщення, функціонал та зовнішній вигляд практично не відрізнятиметься, тому наведення прикладу коду буде лише раз.

Код навігаційної панелі:

```
<nav
        class="navbar navbar-expand-sm navbar-dark bg-dark"
    >
        <div class="container">
            <a class="navbar-brand" href="#">Easy English</a>
            <button
                class="navbar-toggler d-lg-none"
                type="button"
                data-bs-toggle="collapse"
                data-bs-target="#collapsibleNavId"
                aria-controls="collapsibleNavId"
                aria-expanded="false"
                aria-label="Toggle navigation"
                aria-current="page"
                >
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse"
id="collapsibleNavId">
                <ul class="navbar-nav me-auto mt-2 mt-lg-0">
                    <li class="nav-item">
                        <a class="nav-link fs-6 text"
href="./topics_list_page.html" aria-current="page"
                            >Перелік тем
                            <span class="visually-
hidden">(current)</span></a
```

```
                        >
                    </li>
                </ul>
                <div class="flex-shrink-0 dropdown">
                    <a href="#" class="d-block link-dark
text-decoration-none dropdown-toggle" id="dropdownUser2"
data-bs-toggle="dropdown" aria-expanded="false">
                      <img id="nav-profile-picture" src=""
alt="mdo" width="64" height="64" class="rounded-circle">
                    </a>
                    <ul class="dropdown-menu text-small
shadow" aria-labelledby="dropdownUser2">
                      <li><a class="dropdown-item"
href="./user_profile.html">Профіль</a></li>
                      <li><a class="dropdown-item"
href="./profile_settings.html">Налаштування</a></li>
                      <li><hr class="dropdown-divider"></li>
                      <li><a class="dropdown-item"
onclick="clearSession()" href="./login_page.html">Вийти з
профілю</a></li>
                    </ul>
                  </div>

            </div>

        </div>
    </nav>
```

Код особистої інформації користувача, виключаючи елементи як фотографію та текстове поле:

```
                    <img id="profilePicture" class="rounded-
circle mt-5" width="150px" src="/images/female_pic.png"
alt="Profile Picture">
                <span class="text-black-50">Мене звати 11 1
span>\
                <span class="text-black-50">Я Вчитель</span>
```

Код таблиці:

```
<table class="table table-striped">
    <thead>

<tr>
            <th>
                Номер спроби
            </th>
            <th>
                Айді теми
            </th>
            <th>
                Вірних відповідей
            </th>
        </tr>
    </thead>
    <tbody class="table">
    <tr>
        <td>1</td>
        <td>1</td>
        <td>0/6</td>
    </tr>
    </tbody>
</table>
```

Наступним кроком, буде створення сторінки з можливістю налаштувати деяку особисту інформацію, таку як ім’я, прізвище та стать. Також, використовуватиметься навігаційна панель. Виглядатиме ця сторінка як на рисунку 3.4.

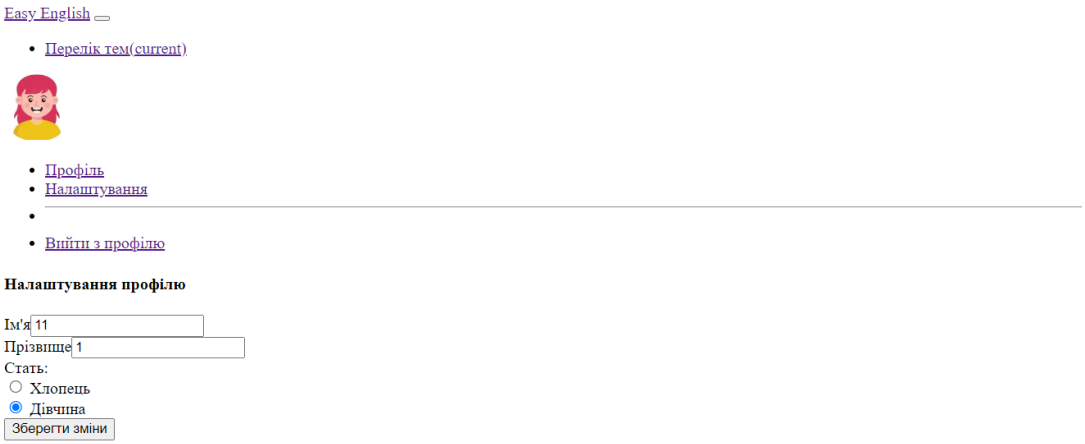

Рисунок 3.4 — Вигляд сторінки «налаштування профілю» з використанням форм

Розміщуватиметься на даній сторінці текстові поля, поля для вводу, зображення та кнопки.

Код полів для вводу:

```
                    <div class="col-md-6"><label
class="labels">Ім'я</label><input type="text"
id="nameforchange" class="form-control" placeholder="first
name" value=""></div>

                    <div class="col-md-6"><label
class="labels">Прізвище</label><input type="text"
id="surnameforchange" class="form-control" value=""
placeholder="surname"></div>
```

Код вибору статі:

```
                        <div class="form-check">

                            Стать:<br>

                            <input class="form-check-input"
type="radio" name="flexRadioDefault1" id="flexRadioDefault11"
value="Хлопець">
```

```
                            <label class="form-check-label"
for="flexRadioDefault11">Хлопець</label>
                        </div>
                        <div class="form-check">
                            <input class="form-check-input"
type="radio" name="flexRadioDefault1" id="flexRadioDefault22"
value="Дівчина">
                            <label class="form-check-label"
for="flexRadioDefault22">Дівчина</label>
                        </div>
```

Код кнопки:

```
                <button class="btn btn-primary profile-
button" type="button" data-bs-toggle="modal" data-bs-
target="#exampleModal">Зберегти зміни</button>
```

Таким чином, користувач в майбутньому матиме змогу змінити, не важливі для функціонування системи, дані самостійно використовуючи дану сторінку.

Для перегляду всіх існуючих тем, потрібно створити відповідну сторінку з переліком. Розміщуватимуться текстові поля, кнопки, зображення, текст з посиланням та навігаційна панель. Виглядатиме сторінка, як на рисунку 3.5.

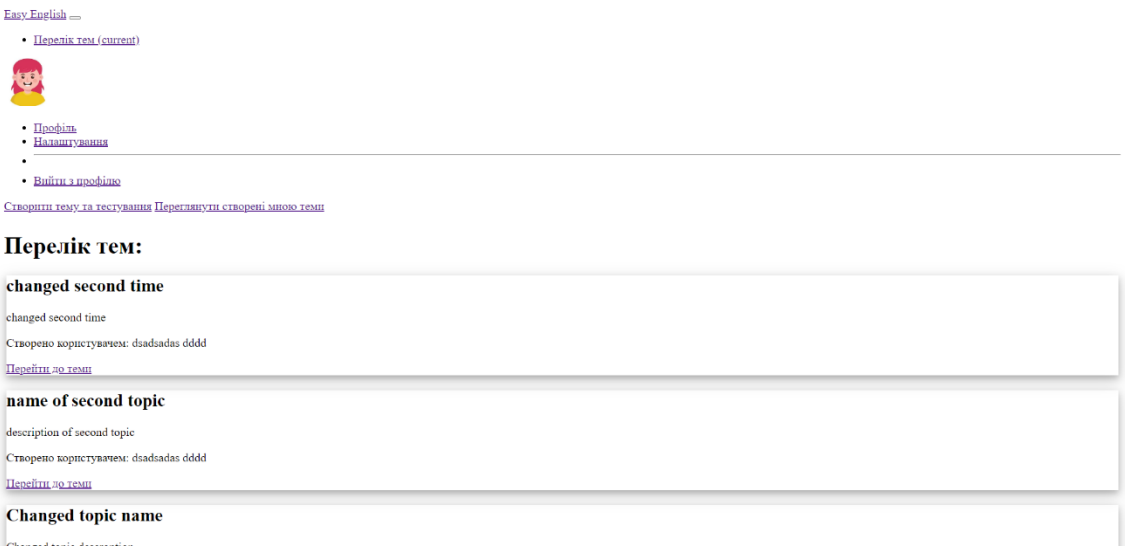

Рисунок 3.5 — Вигляд сторінки «перелік тем» з використанням форм

Код текстового поля:

```
    <h1 class="text-center" style="margin-top: 25px;">Перелік тем:</h1>
```

Наступний продемонстрований код має є частковим, в подальшій розробці буде реалізовано повноцінний код з функціоналом автоматичного виводу всіх тем без необхідності робити цього в ручну адміністраторами системи. Код блоку з інформацією про тему:

```
                    <div class="container">
                        <div style="margin-left: 3px; margin-right: 3px; margin-top: 10px; box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);" class="row py-5">
                            <div class="col-md-9">
                                <div class="lc-block">
                                    <div editable="rich">
                                        <h2><strongназва теми</strong></h2>
                                        <p>підпис теми</p>
                                        <p>Створено користувачем: автор теми</p>
                                    </div>
                                </div>
                            </div>
                            <div class="col-md-3 align-self-center text-center">
                                <div class="lc-block">
                                    <a class="btn btn-link btn-lg" href="./topic_information.html?topicInformationIdайді теми" role="button">Перейти до теми</a>
                                </div>
                            </div>
                        </div>
                    </div>
```

Код кнопок:

```
                    <a style="margin-top: 10px" class="btn
btn-primary" href="./create_topic_and_test_page.html"
role="button">Створити тему та тестування</a>

                    <a style="margin-top: 10px" class="btn
btn-primary" href="./edit_created_topics.html"
role="button">Переглянути створені мною теми</a>
```

Користувач матиме можливість перейти до наступної сторінки клацнувши відповідну кнопку. З можливих: створення теми, перегляд створених користувачем тем та перехід до сторінки з інформацією теми.

Для функціонування системи, потрібна сторінка для створення цих самих тем. Сторінка матиме лише достатню кількість полів що відповідатимуть полям в базі даних та функціональну можливість додавати дані. «створення теми» виглядатиме як на рисунку 3.6.

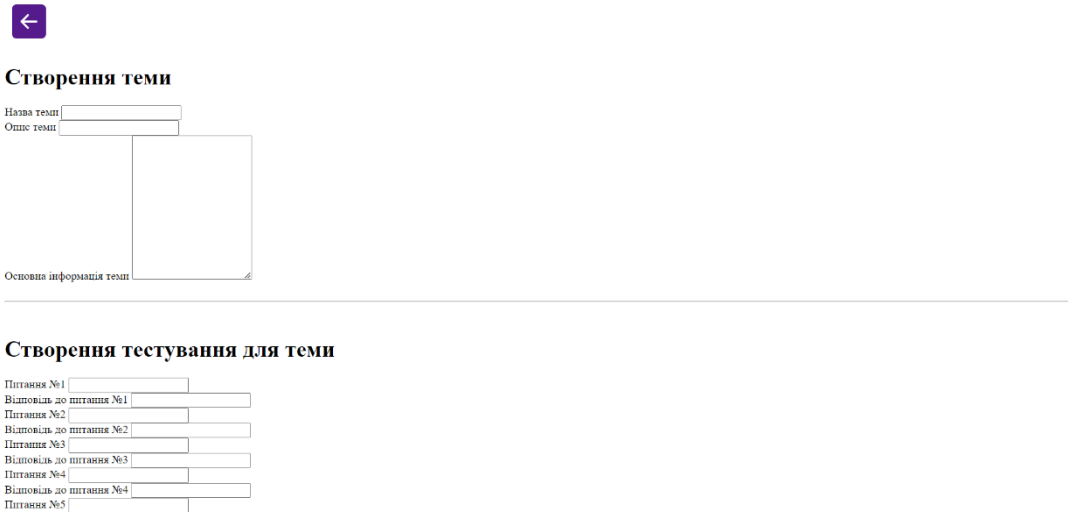

Рисунок 3.6 — Вигляд «створення теми» з використанням форм

Припускається, що користувач може зайти на дану сторінку лише для ознайомлення, та не бажатиме створювати жодної теми. Для цього випадку, буде реалізована кнопка, яка поверне користувача до переліку тем. Код кнопки повернення:

```
<a href="./topics_list_page.html" style="margin: 10px;
display: inline-block;">
    <i class="bi bi-arrow-left-square-fill"></i>
    <svg xmlns="http://www.w3.org/2000/svg" width="48"
height="48" fill="currentColor" class="bi bi-arrow-left-
square-fill" viewBox="0 0 16 16">
        <path d="M16 14a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V2a2 2 0
0 1 2-2h12a2 2 0 0 1 2 2zm-4.5-6.5H5.707l2.147-2.146a.5.5 0 1
0-.708-.708l-3 3a.5.5 0 0 0 0 .708l3 3a.5.5 0 0 0 .708-
.708L5.707 8.5H11.5a.5.5 0 0 0 0-1"/>
</svg>
</a>
```

Для користувача запропоновано всі необхідні полі для заповнення, що будуть використоуватися при виведені інформації. Код полів для створення теми:

```
<h1 class="text-center">Створення теми</h1>
<form id="topicForm">
    <div class="container">
        <div class="row">
          <div class="col-sm">
            <label for="" class="form-label">Назва
теми</label>
            <input type="text" class="form-control" id="form-
topic-name" placeholder=""/>
          </div>
        </div>
        <div class="row">
            <div class="col-sm">
              <label for="" class="form-label">Опис
теми</label>
              <input type="text" class="form-control"
id="form-topic-description" placeholder=""/>
            </div>
```

```
          </div>
          <div class="row">
            <div class="col-sm">
              <label for="" class="form-label">Основна
інформація теми</label>
              <textarea class="form-control" name=""
id="form-topic-main-information" rows="13"></textarea>
          </div>
        </div>
      </div>
```

На цій сторінці, для функціонування, додано можливість створити ще й тестування для відповідної теми. Користувач матиме змогу створити все одразу ж, не переходячи по різним сторінкам. Код полів для створення тестування для теми:

```
<h1 class="text-center">Створення тестування для теми</h1>
      <div class="container">
        <div class="row">
          <div class="col-sm">
            <label for="" class="form-label">Питання
№1</label>
            <input type="text" class="form-control" id="form-
question1" placeholder=""/>
          </div>
          <div class="col-sm">
```

Тестування складатиметься з 6 питань та 6 відповідей, відповідно полів для створення 12. Одним з прикладів реалізації полів для заповнення буде наступний код:

```
            <label for="" class="form-label">Відповідь до
питання №1</label>
            <input type="text" class="form-control" id="form-
answer1" placeholder=""/>
```

Кнопка, яка підтверджуватиме створення теми та тестування, розташовуватиметься знизу. Код кнопки підтвердження:

```
<button type="submit" class="btn btn-primary" data-bs-
toggle="modal" data-bs-target="#exampleModal">Підтвердити
створення теми та тестування</button>
```

Для перегляду створених користувачем тем, призначена сторінка «створені мною теми». Наповнення складається з схожих блоків як в «перелік тем», проте на даній сторінці немає навігаційної панелі. Натомість, наявна кнопка повернення як на попередній сторінці. Виглядатиме сторінка як на рисунку 3.7.

Рисунок 3.7 — Вигляд «створені мною теми» з використанням форм

Кнопка повернення ідентична що і на попередній сторінці, посилання сторінки, на яку переадресовуватиметься користувач ідентичне. Код кнопки повернення можна було переглянути раніше.

Блоки з інформацією про теми практично не відрізняються від блоків з сторінки переліку тем, відмінністю буде лише кнопка, яка перенаправить користувача на сторінку, призначена для оновлення інформації. Код блоків тем:

```
        <div class="container">
        <div style="margin-left: 3px; margin-right: 3px;
margin-top: 10px; box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2),
0 6px 20px 0 rgba(0, 0, 0, 0.19);" class="row py-5">
            <div class="col-md-9">
                <div class="lc-block">
                    <div editable="rich">
```

```
                        <h2><strong>Назва темиstrong></h2>
                        <p>Підпис теми</p>
                    </div>
                </div>
            </div>
            <div class="col-md-3 align-self-center text-
center">
                <div class="lc-block">
                    <a class="btn btn-link btn-lg"
href="./topic_update.html?topicId=айді теми"
role="button">Перейти до редагування теми</a>
                </div>
            </div>
        </div>
    </div>
```

Натиснувши кнопку «перейти до редагування теми», користувач попаде на сторінку редагування. Елементами вона не відрізнятиметься від «створення теми». Різниця відображатиметься лише в функціоналі. Виглядатиме сторінка, як на рисунку 3.8.

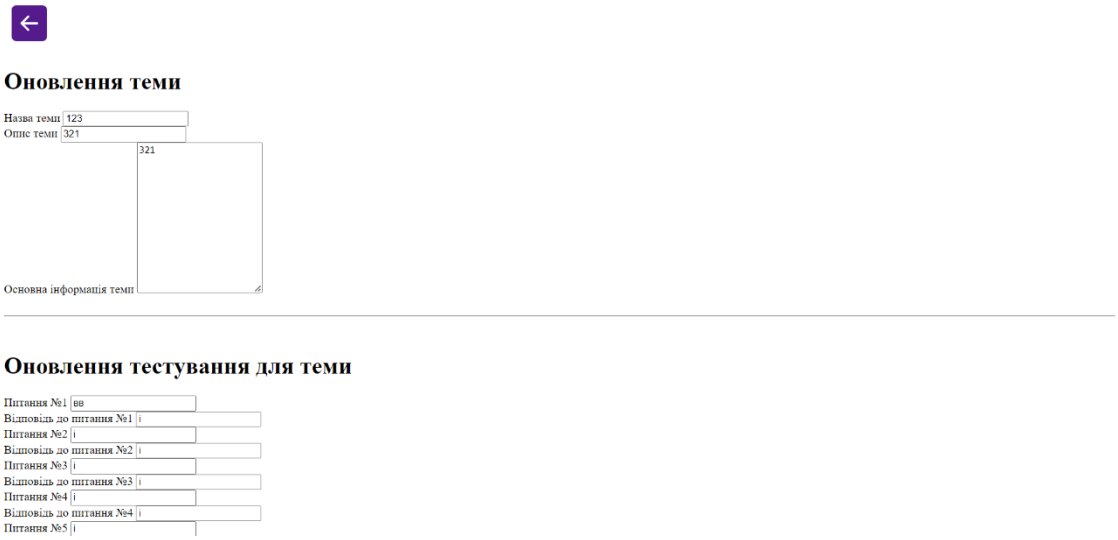

Рисунок 3.8 — Вигляд сторінки «оновлення теми» з використанням форм

На цьому етапі, різниця в коді лише в кнопці. Посилання на сторінку відрізнятиметься і виглядатиме наступним чином:

```
<a href="./edit_created_topics.html" style="margin: 10px;
display: inline-block;">
    <i class="bi bi-arrow-left-square-fill"></i>
    <svg xmlns="http://www.w3.org/2000/svg" width="48"
height="48" fill="currentColor" class="bi bi-arrow-left-
square-fill" viewBox="0 0 16 16">
        <path d="M16 14a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V2a2 2 0
0 1 2-2h12a2 2 0 0 1 2 2zm-4.5-6.5H5.707l2.147-2.146a.5.5 0 1
0-.708-.708l-3 3a.5.5 0 0 0 0 .708l3 3a.5.5 0 0 0 .708-
.708L5.707 8.5H11.5a.5.5 0 0 0 0-1"/>
    </svg>
</a>
```

Якщо користувач захоче переглянути всю інформацію теми, сторінка «інформація теми» відображатиме основну інформацію, назву та підпис теми. Додатково, внизу, відображатиметься кнопка, для переходу на сторінку «тестування». «інформація теми» виглядатиме, як на рисунку 3.9.

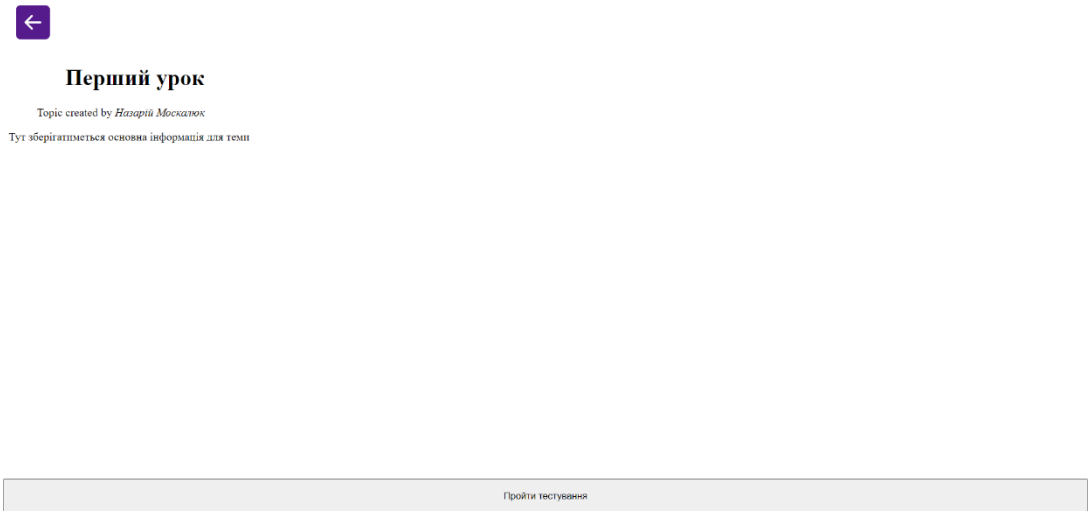

Рисунок 3.9 — вигляд «інформація теми» з використанням форм

Надпис та підпис теми матиме такий код:

```
<figure class="text-center">
            <blockquote class="blockquote">
            <h1>Перший урок</h1>
```

```
</blockquote>
            <figcaption class="blockquote-footer">
            Topic created by <cite title="Source
Title">Назарій Москалюк</cite>
            </figcaption>
            </figure>
```

Основна інформація міститиметься в наступному коді:

```
<div class="container">
            <div class="row">
            <div class="col-1" >
            </div>
            <div class="col-10">
            Тут зберігатиметься основна інформація для тем
            </div>
            <div class="col-1">
            </div>
            </div>
```

Кнопка, яка перенаправлятиме користувача на сторінку для тестування, розташовуватиметься внизу сторінки, та має наступний код:

```
            <button
onclick="location.href='./topic_test_page.html?topicTestId=Ай
ді тесту теми';" class="btn btn-primary"
style="position:fixed;bottom:0;left:0;right:0;height:50px;"
type="button">Пройти тестування</button>
```

Сторінка «тестування», матиме лише поля для вводу, текстові поля та кнопки. Повернутися користувач може лише після завершення тестування. Виглядатиме сторінка, як на рисунку 3.10.

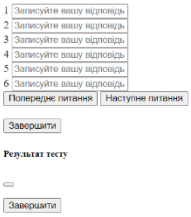

Рисунок 3.10 — Вигляд «тестування» з використанням форм

Сторінка матиме код для 6 полів для вводу та 6 текстових полів, відповідно для кожного питання:

```
    <div class="mb-3" id="question1Div">
        <label for="question1" class="form-label1"
id="labelforquestion1"></label>
        <input type="text" class="form-control"
name="question1" id="question1" aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)"/>
        </div>

        <div class="mb-3" id="question2Div">
        <label for="question1" class="form-label2"
id="labelforquestion2"></label>
        <input type="text" class="form-control"
name="question2" id="question2" aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)" />
    </div>

        <div class="mb-3" id="question3Div">
        <label for="question1" class="form-label3"
id="labelforquestion3"></label>
```

```
        <input type="text" class="form-control"
name="question3" id="question3" aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)" />
    </div>

        <div class="mb-3" id="question4Div">
        <label for="question1" class="form-label4"
id="labelforquestion4"></label>
        <input type="text" class="form-control"
name="question4" id="question4"  aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)" />
    </div>

        <div class="mb-3" id="question5Div">
        <label for="question1" class="form-label5"
id="labelforquestion5"></label>
        <input type="text" class="form-control"
name="question5" id="question5" aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)" />
    </div>

        <div class="mb-3" id="question6Div">
        <label for="question1" class="form-label6"
id="labelforquestion6"></label>
        <input type="text" class="form-control"
name="question6" id="question6" aria-describedby="helpId"
placeholder="Записуйте вашу відповідь сюди :)" />
    </div>
```

Для подальшої реалізації функціоналу, створені кнопки перемикання питань:

```
        <button type="button" class="btn btn-primary"
id="prevQuestion">Попереднє питання</button>

        <button type="button" class="btn btn-primary"
id="nextQuestion">Наступне питання</button>
```

Кнопка, яка завершуватиме тестування та переадресовуватиме користувача матиме наступний код:

```
    <button type="button" class="btn btn-primary" data-bs-
toggle="modal" data-bs-target="#exampleModal">

        Завершити

    </button>
```

Наступним етапом реалізації інтерфейсу буде дизайн. Буде використовуватись Bootstrap 5, який спрощує весь процес та додає можливість зручно переглядати систему на мобільних девайсах. Далі, буде продемонстровано рисунки сторінок вже з готовим дизайном.

Розпочнемо з початкової сторінки. Вхід виглядатиме, як на рисунку 3.11.

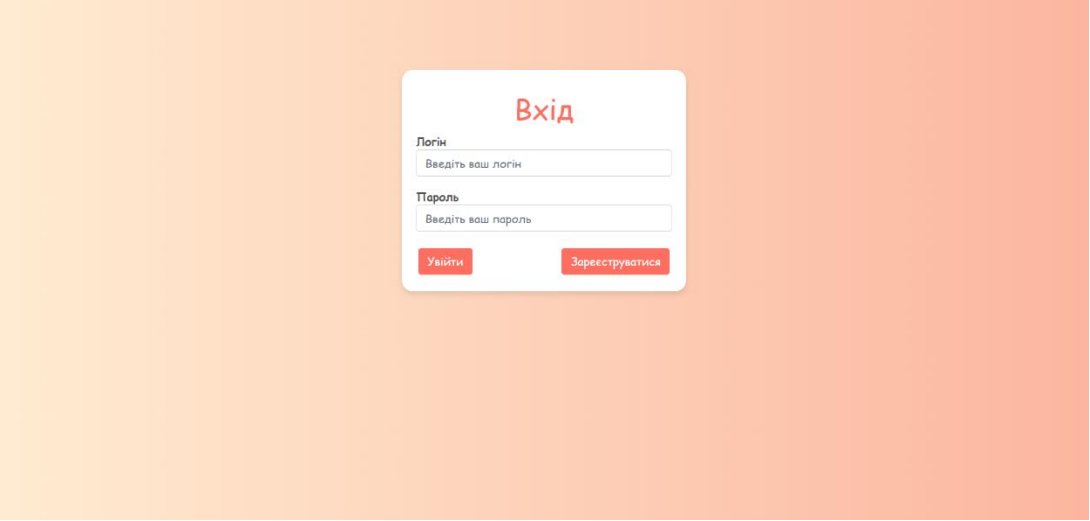

Рисунок 3.11 — Сторінка входу з дизайном

Сторінка входу на мобільних девайсах виглядатиме як на рисунку 3.12.

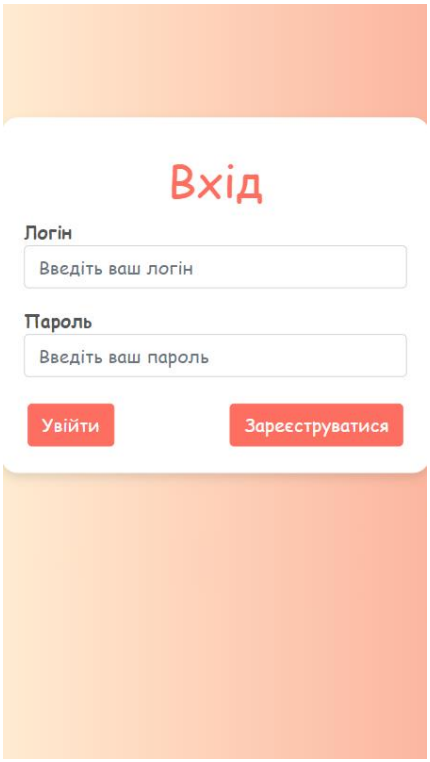

Рисунок 3.12 — Вигляд мобільної адаптації сторінки входу

Реєстрація матиме вигляд, як на рисунку 3.13.

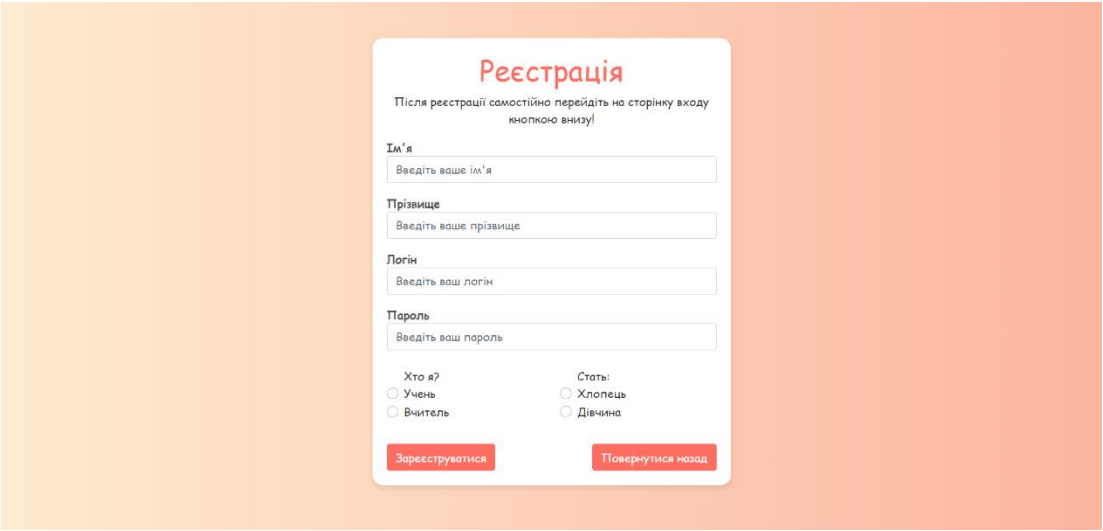

Рисунок 3.13 — Сторінка реєстрації з дизайном

Сторінка реєстрації на мобільних девайсах виглядатиме наступним чином (рисунок 3.14).

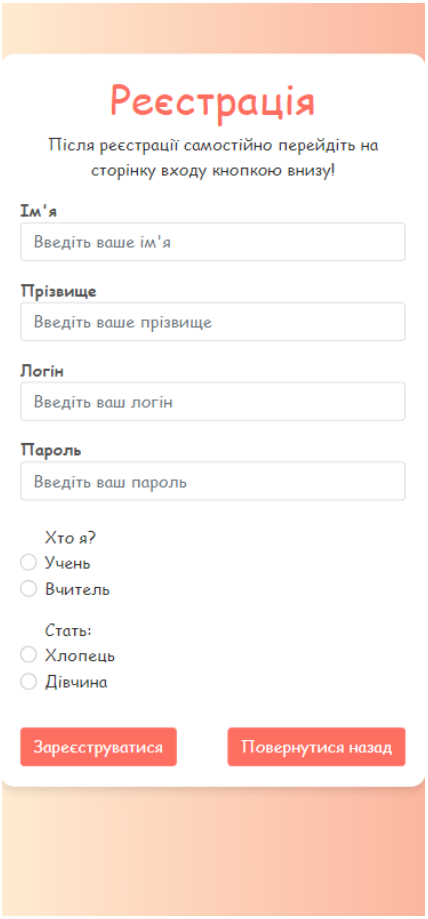

Рисунок 3.14 — Сторінка реєстрації на мобільних девайсах1

Після входу, користувач побачить особистий профіль (рисунок 3.15), на мобільних девайсах, як на рисунку 3.16.

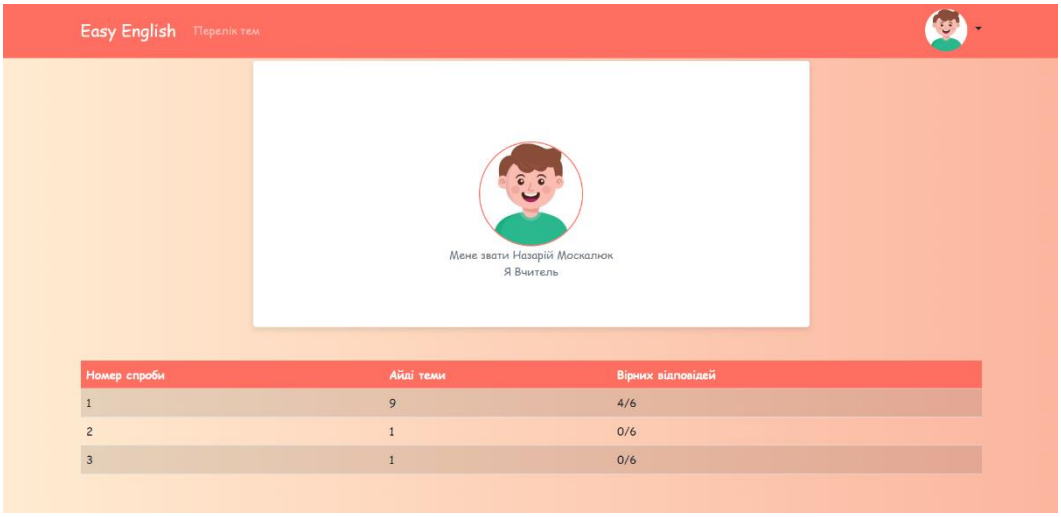

Рисунок 3.15 — Сторінка особистого профілю з дизайном

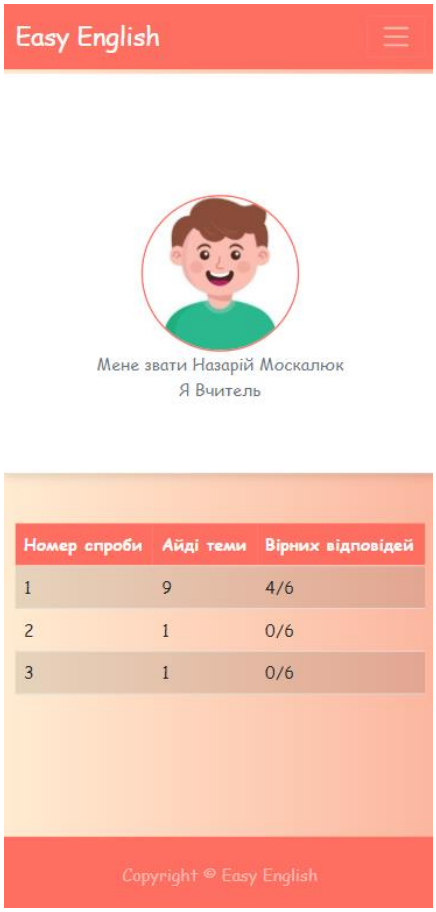

Рисунок 3.16 — Сторінка особистого профілю на мобільних девайсах

Варто зауважити, що Bootstrap 5, дозволяє створювати навігаційні панелі, що на мобільних девайсах будуть виглядати зовсім по іншому, як можна побачити на рисунку 3.16. В відкритому вигляді, елемент виглядатиме як на рисунку 3.17.

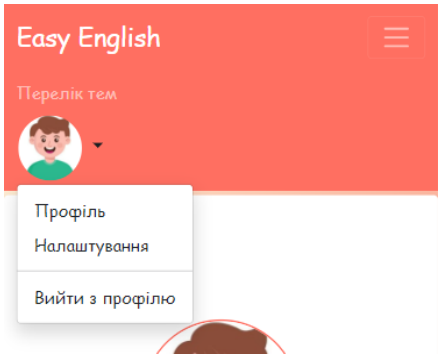

Рисунок 3.17 — Розгорнутий вигляд навігаційної панелі на мобільних девайсах

Налаштування особистої інформації виглядатиме наступним чином (рисунок 3.18 та рисунок 3.19) .

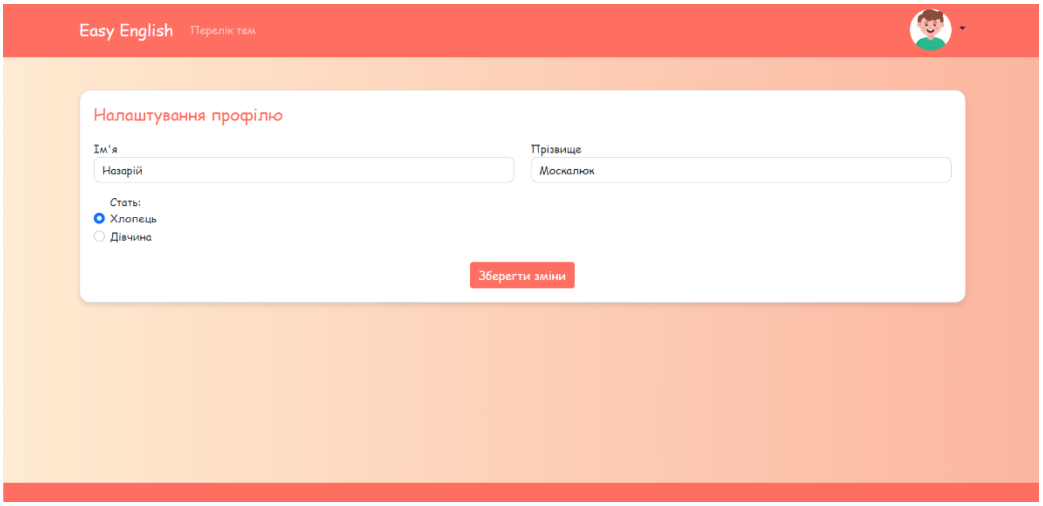

Рисунок 3.18 — Сторінка налаштувань особистої інформації з дизайном

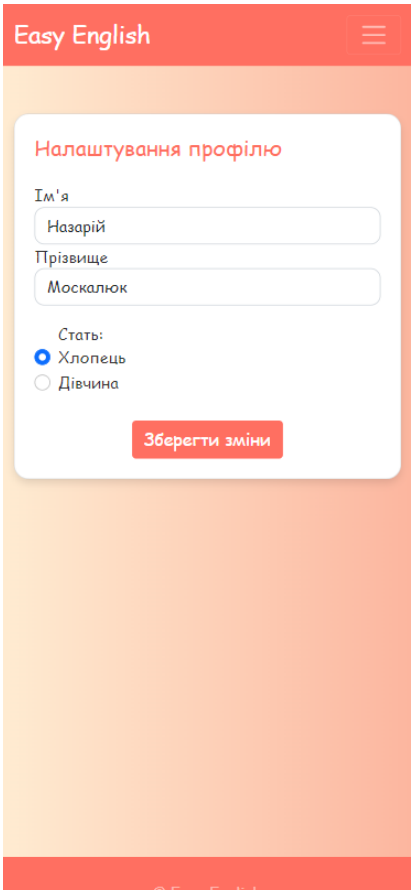

Рисунок 3.19 — Сторінка налаштувань особистої інформації на мобільних девайсах

Перелік тем, не повинен мати відразливого дизайну, тому виглядатиме саме так, як на рисунку 3.20 та 3.21.

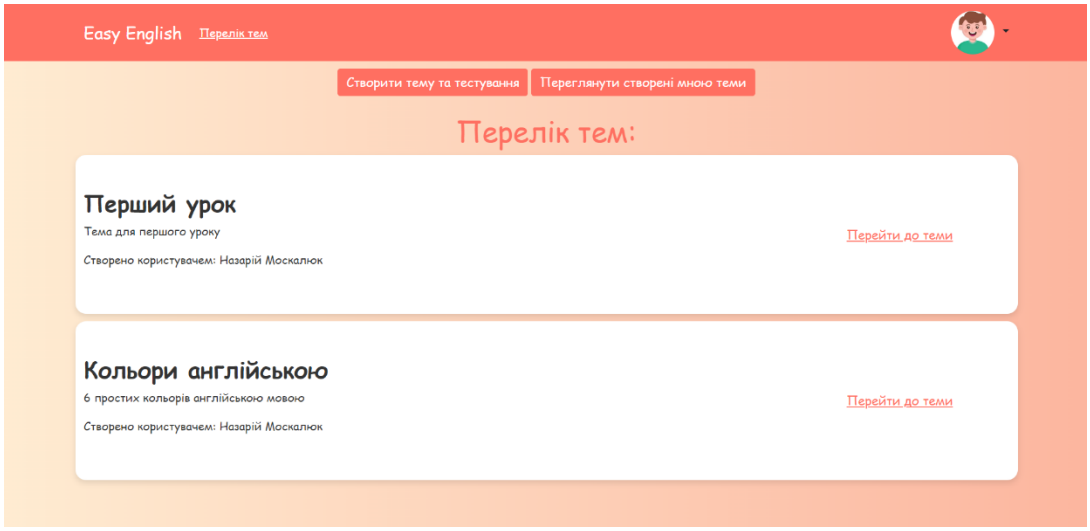

Рисунок 3.20 — Сторінка переліку тем з дизайном

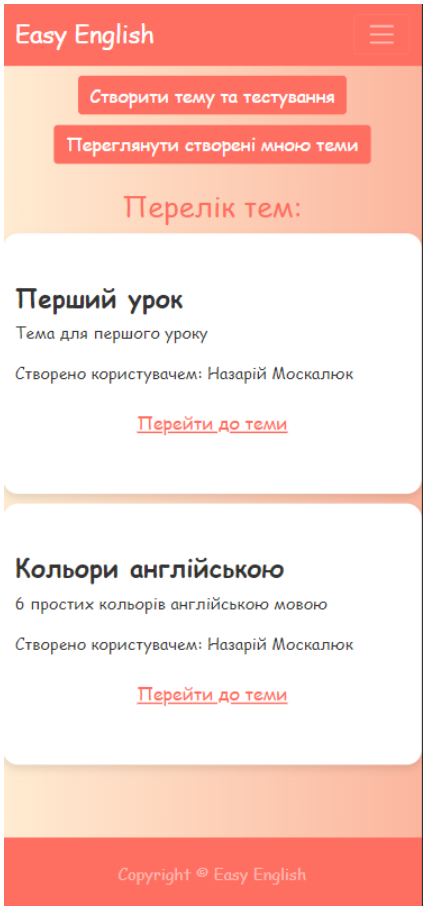

Рисунок 3.21 — Сторінка переліку тем на мобільних девайсах

Сторінка створення теми, повинна бути максимально приємною та без зайвих деталей, через те, що користувач буде краще сконцентрований. Виглядатиме наступним чином (рисунок 3.22 та рисунок 3.23).

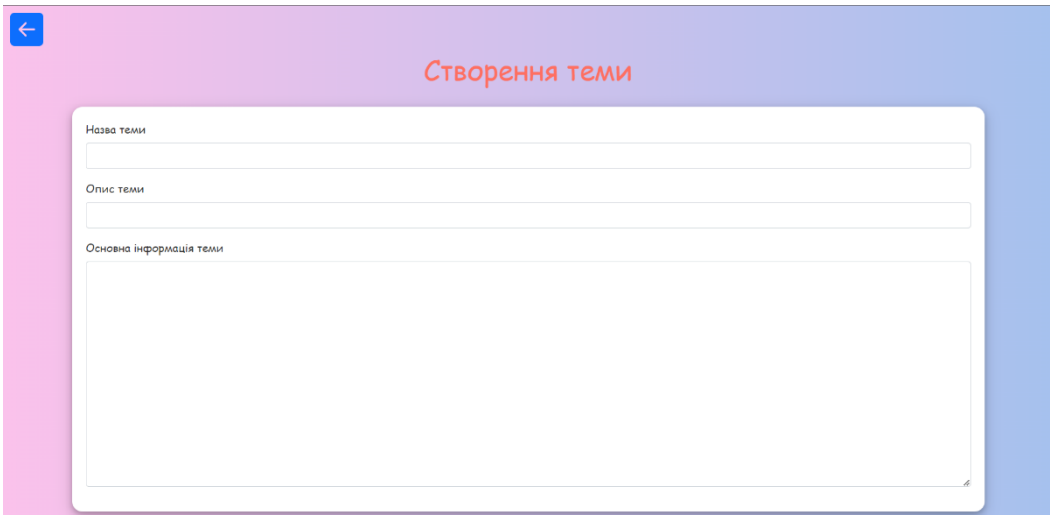

Рисунок 3.22 — Сторінка створення теми та тестування з дизайном

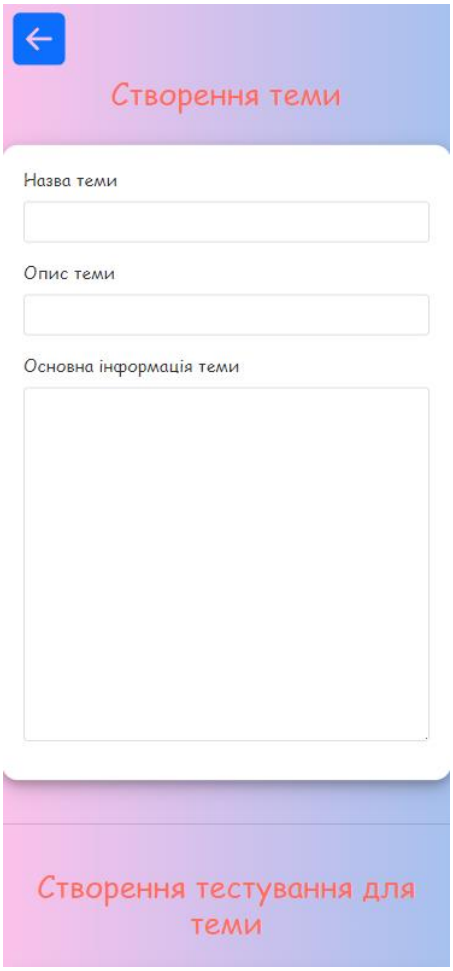

Рисунок 3.23 — Сторінка створення теми а тестування на мобільних девайсах

Перегляд створених користувачем тем, буде схожим до двох попередніх сторінок як відсутністю навігаційної панелі так і виглядом переліку тем, та виглядатиме як на рисунок 3.24 та рисунок 3.25.

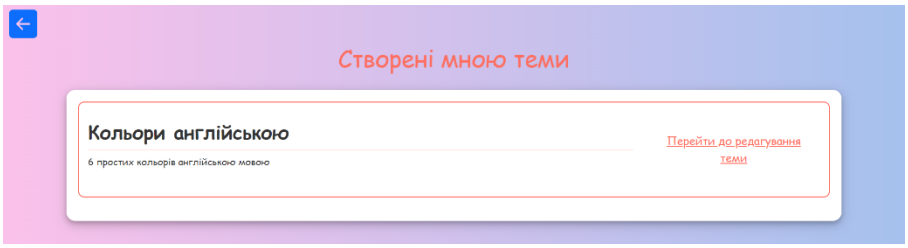

Рисунок 3.24 — Сторінка створених користувачем тем з дизайном

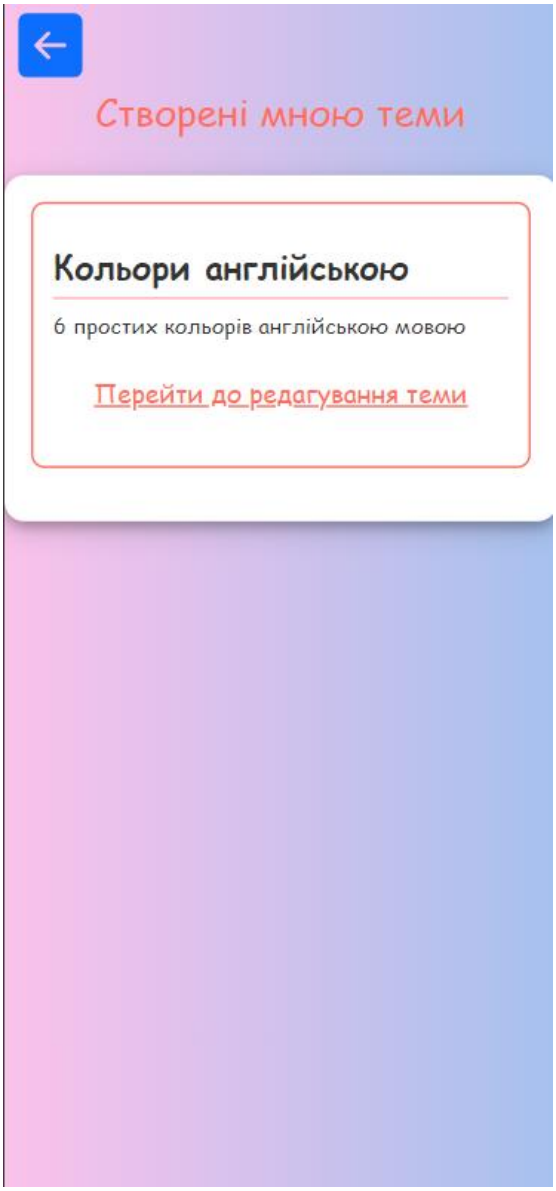

Рисунок 3.25 — Сторінка створених користувачем тем на мобільних девайсах

Редагування тем, виглядатиме дуже схоже до «створення тем», а саме як на рисунку 3.26 та рисунку 3.27.

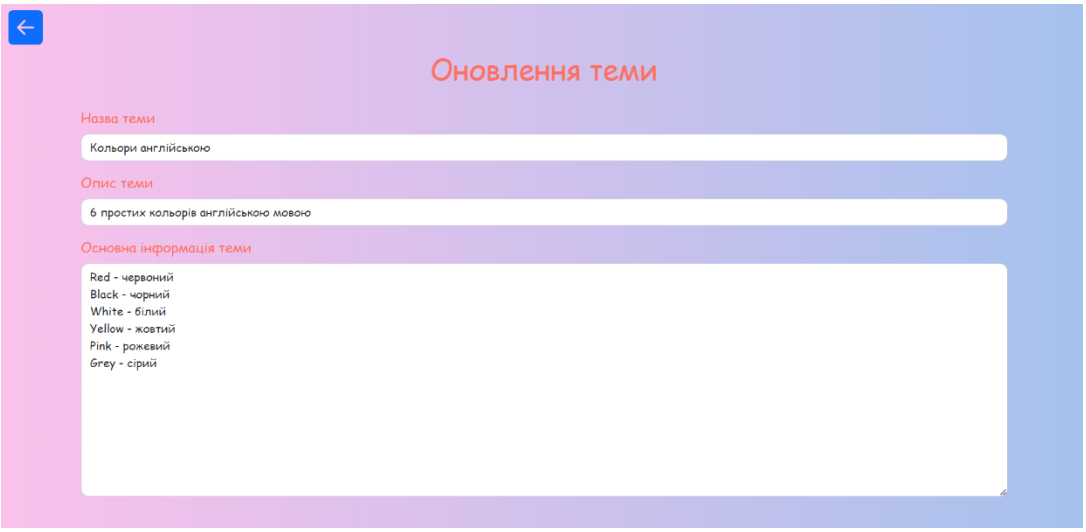

Рисунок 3.26 — Сторінка оновлення теми з дизайном

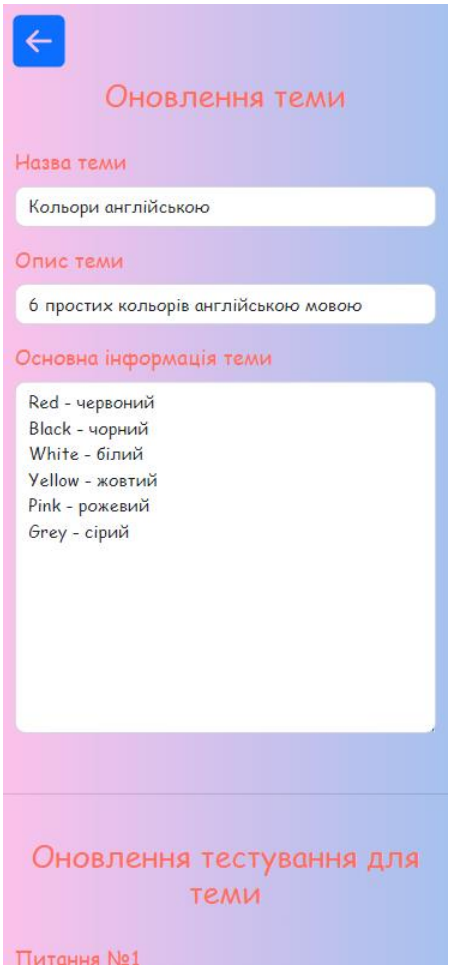

Рисунок 3.27 — Сторінка оновлення теми на мобільних девайсах

«Перегляд інформації теми» теми не повинен містити зайвих елементів, тому виглядатиме як на рисунку 3.28 на рисунку 3.29.

Рисунок 3.28 — Сторінка перегляду інформації теми з дизайном

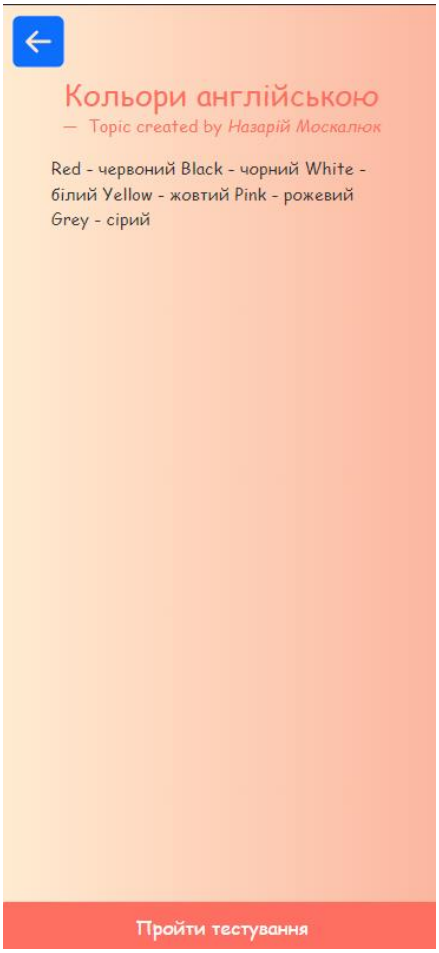

Рисунок 3.29 — Сторінка перегляду інформації теми на мобільних девайсах

Сторінка для проходження тестування виглядатиме, як на рисунку 3.30 та рисунку 3.31.

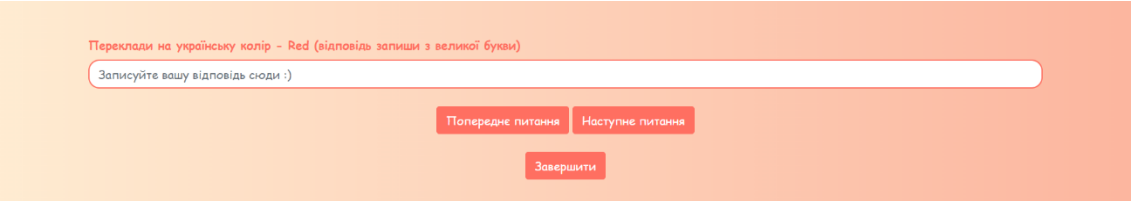

Рисунок 3.30 — Сторінка проходження тестування з дизайном

Рисунок 3.31 — Сторінка проходження
тестування на мобільних девайсах

Підсумовуючи даний етап, всі сторінки системи містять необхідні поля, кнопки, елементи та дизайн. Вигляд приємний та строгий.

3.2 Реалізація функціоналу

Реалізація функціональної частини системи вимагає використання різноманітних програмних засобів. Бібліотеки та фреймворки спрощують процес написання коду в рази, а подекуди і зовсім надають можливості, що не доступні без використання додаткових модулів.

Для коректної роботи, даній системі потрібно завантажити бібліотеку node.js. Node.js — це середовище виконання JavaScript, яке дозволяє запускати код JavaScript поза браузером. Node.js використовує асинхронну модель, керовану подіями, і може обробляти багато запитів одночасно, що робить його дуже ефективним під час створення високонавантажених серверних програм.

На основі механізму Google V8 Node.js досягає високої продуктивності завдяки компіляції коду JavaScript безпосередньо в машинний код.

Express.js — це мінімальний і гнучкий фреймворк для Node.js, який спрощує процес розробки серверних програм. Express.js надає потужний набір інструментів для створення веб-додатків і API. Express.js дозволяє розробникам легко налаштовувати маршрутизацію запитів, обробку проміжного програмного забезпечення та керування станом сесії.

Простота та розширюваність Express.js дозволяє швидко створювати масштабовані серверні програми з чистою та зрозумілою структурою коду.

Для встановлення Node.js та Express.js, необхідно, зайти в термінал будь якому середовищі програмування написати наступні команди: для node.js npm install nodejs==0.1.1 , для express.js npm install express. Після чого, в файлах проекту з’являться файли конфігурації, які містять всю необхідну інформацію для початку роботи.

Спершу, на кожній сторінці додамо можливість переміщуватись між собою. JavaScript дозволяє додати до об’єктів такий функціонал за допомогою рядку:

```
window.location.href = 'назва файлу.html';
```

Підготовчим етапом до під’єднання бази даних коду, буде налаштування роботи сесій. Існує два види, LocalStorage та SessionStorage.

В localstorage, дані зберігаються постійно, навіть після закриття браузера, і доступні у всіх вкладках та вікнах для даного домену. Використовується для зберігання довготривалих даних, таких як налаштування користувача або токени авторизації. В sessionstorage, дані зберігаються лише протягом поточної сесії і видаляються після закриття вкладки або вікна браузера. Кожна вкладка має своє окреме сховище, подібне до localStorage за обсягом. Використовується для зберігання тимчасових даних, таких як стан форми або налаштування, що не повинні залишатися після завершення сесії користувача.

Надання значень сесії, в даній системі, відбуватиметься лише при вході на відповідній сторінці. Далі, наведений код, який набуває дані після отримання та підтвердження, які будуть реалізовані в підрозділі бази даних:

```
sessionStorage.setItem("userId", user_id);
sessionStorage.setItem("userName", user_name);
sessionStorage.setItem("userSurname", user_surname);
sessionStorage.setItem("userGender", user_gender);
sessionStorage.setItem("userLogin", user_login);
sessionStorage.setItem("userPassword", user_password);
sessionStorage.setItem("userPermitionLevel",
user_permition_level);
```

Отримати дані, можна за допомогою призначення окремій змінній, значення, яке зберігається під тим чи іншим ім’ям:

```
const user_Id = sessionStorage.getItem("userId");
```

Очистити дані можна за допомогою простої команди removeItem, як в наступному коді:

```
sessionStorage.removeItem("userId");
```

Для реєстрації та входу користувача, будуть використовуватись асинхронні запити до бази даних. Реалізації складається з двох частин: запит до методу та запит методу до бази даних. Оскільки база даних ще не готова, розглянемо перший. Реєстрація, передбачає собою збір даних з полів для вводу, перевірку та надсилання в базу даних. Для того, щоб зберегти дані, вписані користувачем в змінні, можна використати наступний приклад:

```
var accName = document.getElementById("accName").value;
var accSurname = document.getElementById("accSurname").value;
var accLogin = document.getElementById("accLogin").value;
var accPassword
=document.getElementById("accPassword").value;
```

З radio button, інший принцип, потрібно вибрати активне значення використовуючи:

```
var accSelectedGender =
document.querySelector('input[name="flexRadioDefault1"]:check
ed');
```

Структура надсилання запитів до методу дуже схожа. Єдині відмінності можуть бути в кількості надісланих змінних, назві методу та дії, що передують після надсилання. Прикладом, можна навести надсилання реєстраційних даних на посилання http://localhost:3000/register. Дане посилання є частиною синтаксису при звертанні до створеного, за допомогою express, локального запиту. Також, не варто забувати про дані, які потрібно внести у окремий список, для прикладу body. Разом, код виглядатиме наступним чином:

```
var body = {
    user_name: accName,
    user_surname: accSurname,
    user_gender: accSelectedGender.value,
    user_login: accLogin,
    user_password: accPassword,

user_permition_level: accPermitionLevel
```

```
            };

            fetch('http://localhost:3000/register', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(body)
            })
```

В деяких випадках, поставлене завдання вимагає обробки зворотніх даних. Для цього існує метод .then, який викликається після fetch, основне завдання якого опрацювати змінні або дані багатьох типів, які були надіслані назад. Як приклад, візьмем задавання змінним в сесії значень, отриманих при вході на відповідній сторінці:

```
.then(response => response.text())
            .then(data => {
                console.log(data);
                const userData = JSON.parse(data);
                const { user_id, user_name, user_surname,
user_gender, user_login, user_password, user_permition_level
} = userData;

                // Set session data
                console.log(sessionStorage.getItem("userName"
));
                sessionStorage.setItem("userId", user_id);
                sessionStorage.setItem("userName",
user_name);
                sessionStorage.setItem("userSurname",
user_surname);
                sessionStorage.setItem("userGender",
user_gender);
```

```
                sessionStorage.setItem("userLogin",
user_login);
                sessionStorage.setItem("userPassword",
user_password);
                sessionStorage.setItem("userPermitionLevel",
user_permition_level);
            });
        }
```

На навігаційній панелі повинна бути одна з двох фотографій як на рисунку 3.32 та рисунку 3.33. В залежності від статі буде відображатись відповідна фотографія.

Рисунок 3.32 — Фотографія намальованої дівчини

Рисунок 3.33 — Фотографія намальованого хлопця

Для того, щоб перевірити, яка стать в користувача, реалізовано наступний код:

```
                sessionStorage.setItem("userGender",
user_gender);
                let tempVariableForPicture;


                if (user_gender === "Хлопець") {
                    tempVariableForPicture =
"/images/male_pic.jpg";
                } else if (user_gender === "Дівчина") {
                    tempVariableForPicture =
"/images/female_pic.jpg";
                }


                sessionStorage.setItem("profilePicturePath",
tempVariableForPicture);
```

Тепер, в змінній profilePicturePath, збережено текст з шляхом до фотографії.

На самій навігаційній панелі та в особистому профілі, картика відображатиметься за допомогою наступного коду:

```
<img id="profilePicture" class="rounded-circle mt-5"
width="150px" src="${profile_Picture_Path}" alt="Profile
Picture">
```

та

```
    document.getElementById("nav-profile-picture").src =
profile_Picture_Path;
```

На сторінці профілю, також є таблиця в якій відображається інформація про спроби проходження тестувань. Для реалізації автоматичного оновлення даних, можна використати запит fetch, приклад якого вже був продемонстрований раніше та написати наступний код:

```
        var table = document.querySelector('.table tbody');


        table.innerHTML = '';
```

```
        var attempt_number = 1;

        data.forEach(rowData => {
            var row = table.insertRow();
            row.insertCell().textContent = attempt_number++;
            row.insertCell().textContent = rowData.topic_test_id;
            row.insertCell().textContent = rowData.attempt_result
    + "/6";
        });
```

На сторінці тестування, потрібно додати декілька строчок коду. Реалізовується такі методи для запобігання одночасного відображення шести питань одночасно. Натомість, користувач бачитиме лише одне питання за раз. Перемкнути на наступне питання можна буде за допомогою кнопок, які з методом матимуть наступний код:

```
        var answer1 = document.getElementById('question1').value;
        var answer2 = document.getElementById('question2').value;
        var answer3 = document.getElementById('question3').value;
        var answer4 = document.getElementById('question4').value;
        var answer5 = document.getElementById('question5').value;
        var answer6 = document.getElementById('question6').value;

        var correctAnswers = 0;

        if (answer1.toLowerCase() ===
    topic_answer_one.toLowerCase()) {
            correctAnswers++;
        }
```

```
if (answer2.toLowerCase() === topic_answer_two.toLowerCase())
{
        correctAnswers++;
    }

    if (answer3.toLowerCase() ===
topic_answer_three.toLowerCase()) {
        correctAnswers++;
    }

    if (answer4.toLowerCase() ===
topic_answer_four.toLowerCase()) {
        correctAnswers++;
    }

    if (answer5.toLowerCase() ===
topic_answer_five.toLowerCase()) {
        correctAnswers++;
    }

    if (answer6.toLowerCase() ===
topic_answer_six.toLowerCase()) {
        correctAnswers++;
    }

    var resultMessage = 'Кількість правильних відповідей: ' +
correctAnswers;
    document.getElementById('resultMessage').innerHTML =
resultMessage;

    document.getElementById('question1').value = '';
    document.getElementById('question2').value = '';
    document.getElementById('question3').value = '';
```

```
    var modal = new
bootstrap.Modal(document.getElementById('exampleModal'));
    modal.show();
```

та

```
document.getElementById('prevQuestion').addEventListener('cli
ck', function() {
    if (currentQuestion > 1) {
        currentQuestion--;
        showQuestion(currentQuestion);
    }
});

document.getElementById('nextQuestion').addEventListener('cli
ck', function() {
    if (currentQuestion < 6) { // Ваша кількість питань
        currentQuestion++;
        showQuestion(currentQuestion);
    }
});



function showQuestion(questionNumber) {
    document.querySelectorAll('.mb-
3').forEach(function(question) {
        question.style.display = 'none';
    });
    document.querySelector('#question' + questionNumber +
'Div').style.display = 'block';
}

showQuestion(currentQuestion);
```

```
function redirectToIndex() {
    window.location.href = './topics_list_page.html';
}
```

В результаті, сторінка тестування набула наступного вигляду (рисунок 3.34).

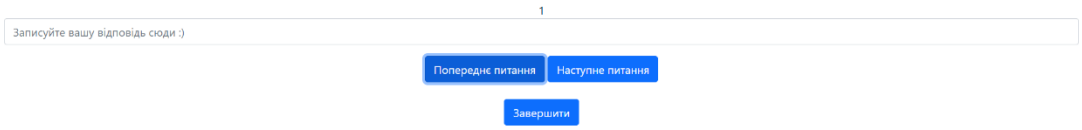

Рисунок 3.34 — Кінцевий вигляд сторінки тестування

Для запобігання перебування користувача на будь якій сторінці кім реєстрації та входу, без попереднього входу, реалізована перевірка на наявність в sessionStorage даних про користувача:

```
if (sessionStorage.getItem("userId") === null ||
sessionStorage.getItem("userId") === undefined) {
    window.location.href = "./login_page.html";
}
```

Код реалізовано на всіх сторінках, крім двох вище сказаних.

3.3 Реалізація бази даних

Використання бази даних потребує додаткових програмних та технічних забезпечень. У випадку реалізації даної системи, фізичного збереження даних не

обов’язкове. Дані будуть зберігатися в реляційній базі даних. Метою даного етапу буде створення та нормалізація полів та зв’язків між таблицями.

Протягом цього етапу, для функціонування бази даних, потрібно встановити додаткове програмне забезпечення під назвою XAMPP (рисунок 3.35). XAMPP — це безкоштовний багатоплатформний стек програмного забезпечення, який включає Apache, MySQL і PHP і використовується для швидкого налаштування локального веб-сервера. Також інтегрована система керування базами даних MySQL.

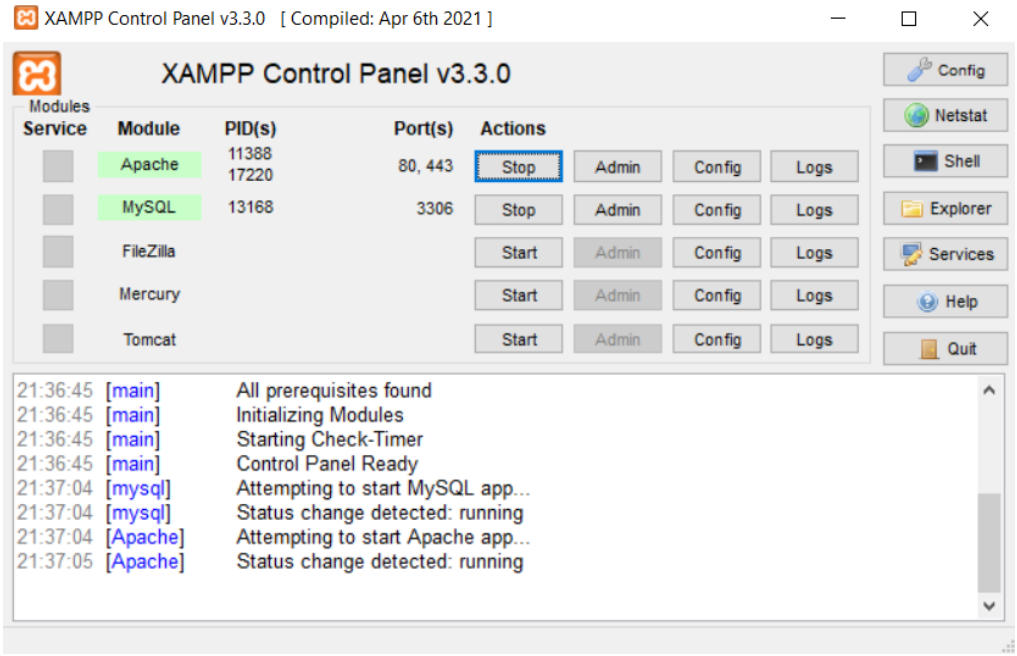

Рисунок 3.35 — Вигляд програми XAMPP

Щоб запустити Apache та MySQL, потрібно на панелі керування (рисунок 3.35) нажати на «Start». Це дозволяє розробникам легко тестувати та налагоджувати веб-програми на своїх комп’ютерах без підключення до Інтернету.

В програмному коді, для підключення проекту до бази даних, використовується express:

```
const express = require('express');
const mysql = require('mysql');
```

```
const app = express();
const port = 3000;



// Create connection to MySQL database
const db_connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'eenglish_database'
});

// Start server
app.listen(port, () => {
    console.log(`Server started on port ${port}`);
});
```

Реалізовуючи функціонал, потрібно додати до необхідних скриптів запити до бази даних. Як приклад, візьмемо запит, який перевіряє правдивість введених користувачем даних, та повертає необхідні дані, щоб скрипт розумів результат. Код продемонстровано далі:

```
app.post('/login', (req, res) => {
    const { user_login, user_password } = req.body;

    // Check if user_login is unique
    const checkUniqueLoginQuery = `SELECT COUNT(*) AS count
FROM user_information WHERE user_login = '${user_login}'`;
    db_connection.query(checkUniqueLoginQuery, (err, result)
=> {
        if (err) {
```

```
            console.error("Error checking unique login:",
    err);
            res.status(500).send('Error registering user');
            return;
        }



        const isLoginUnique = result[0].count === 0;
        if (!isLoginUnique) {
            const sql2 = `SELECT user_password AS
    user_password_from_db FROM user_information WHERE user_login
    = '${user_login}'`;
            db_connection.query(sql2, (err1, result1) => {
                if (err1) {
                    console.error('Error:', err1);
                    res.status(500).send('Error!');
                    return;
                }
                if (result1[0].user_password_from_db ==
    user_password){
                    let is_password_correct = true;
                    res.send(is_password_correct);
                } else {
                    console.log("Something is incorrect!");
                    is_password_correct = true;
                }
            });
        } else {
            console.error('Login is not registered!');
            res.status(400).send('Login is not registered!');
        }


    });
```

```
});
```

Всі запити, включаючи перевірку входу, зберігаються в файлі app.js. Весь код можна переглянути в Додатку А.

3.4 Тестування системи

Під час реалізації системи, останнім етапом є тестування, яке дає змогу своєчасно виявити помилки та несправності, які можуть виникнути в користувача.

Обов'язковими процесами для перевірки є функціонал, без якого повністю або частково неможливе функціонування всіх інших процесів.

Вхід та реєстрація користувача, є одними з найголовніших і найважливіших етапів. Без їх коректної роботи, користувач не матиме змогу увійти та отримати доступ до функціоналу системи. Розпочавши з сторінки реєстрації, вхідними даними будуть:

- ім'я: Віталік;
- прізвище: Іванович;
- логін: ivanovychvit334;
- пароль: vitalik992003;
- учень чи вчитель: учень;
- стать: хлопець.

Очікуваним результатом буде наявність введених даних у базі даних. Як результат, на рисунку 3.36, можна побачити, що система зареєструвала нового користувача з відповідними даними.

user_id	user_name	user_surname	user_gender	user_login	user_password	user_permition_level
148	11	1	Хлопець	1	1	2
149	Назарій	Москалюк	Хлопець	mylogin	mypassword	2
150	Віталік	Іванович	Хлопець	ivanovychvit334	vitalik992003	1

Рисунок 3.36 — Результат тестування реєстрації нового користувача

Тестування входу користувача вимагає в результаті переадресувати користувача на сторінку особистого профілю. Даними для вводу будуть:

– логін: ivanovychvit334;

– пароль: vitalik992003.

В результаті, користувач отримав доступ до профілю, як і очікувалось (рисунок 3.37).

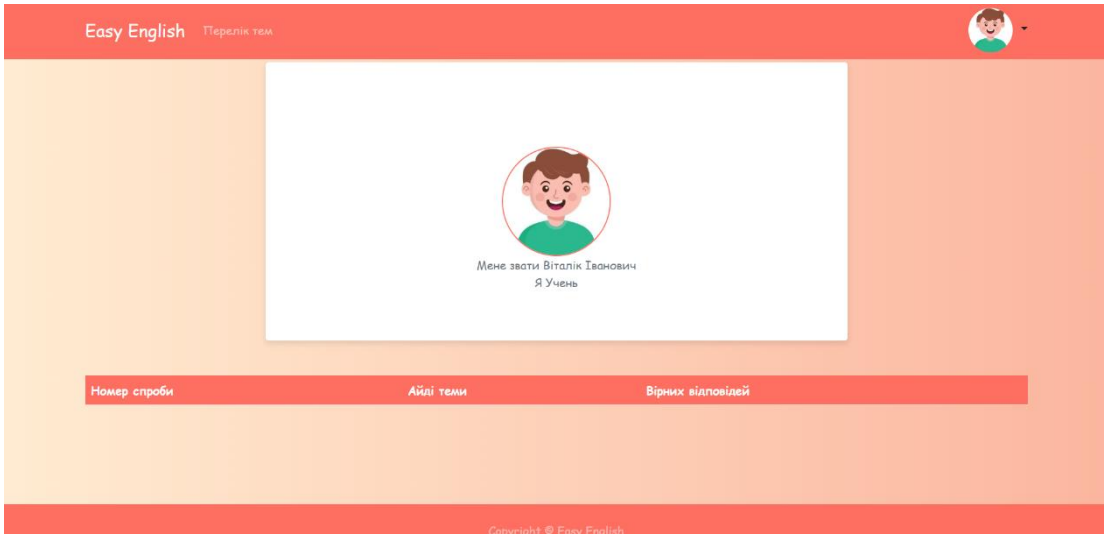

Рисунок 3.37 — Результат тестування входу користувача

Наступним кроком буде перевірка правильності виконання роботи сторінки, що призначена змінювати деякі особисті дані користувача. До прикладу, як перевірку даного функціоналу, змінимо ім’я з Віталік на Артем. Як результат, на сторінці профілю (рисунок 3.38) та в базі даних (рисунок 3.39), користувацьке ім’я змінено.

user_id	user_name	user_surname	user_gender	user_login	user_password	user_permition_level
148	11	1	Хлопець	1	1	2
149	Назарій	Москалюк	Хлопець	mylogin	mypassword	2
150	Артем	Іванович	Хлопець	ivanovychvit334	vitalik992003	1

Рисунок 3.39 — Результат тестування з налаштування особистого профілю в базі даних

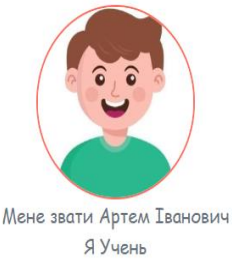

Рисунок 3.38 — Результат тестування з налаштування особистого профілю на сторінці профілю

Увійшовши вчителем, на сторінці «перелік тем», проявляється можливість створити авторську тему та тестування. Створення, є важливим етапом в функціонуванні системи. Як приклад, назвемо тему «Кольори англійською», опис та основну інформацію заповнимо простими реченнями. В поля з питаннями та відповідями, напишемо як і питання на переклад з англійської на українську, так і навпаки. Як результат, тема та тестування створено, та відображено на сторінці (рисунок 3.39).

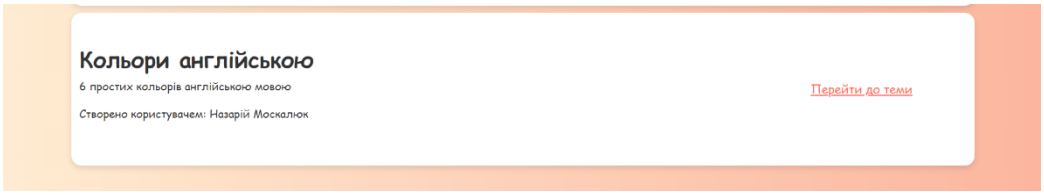

Рисунок 3.39 — Відображення теми в списку

Проходження тестування та отримання результату на основі кількості правильних відповідей є одним з основних функціональних елементів даної системи. Перевірка правильності виконання всіх команд та функцій дуже важлива, адже кінцевий користувач зацікавлений в використанні добре протестованої системи. Як приклад візьмемо раніше створену тему, та дамо правильні відповіді на деякі запитання. Як результат, очікується вивід та внесення даних, про те, що кількість правильних відповідей дорівнює чотирьом (рисунок 3.40).

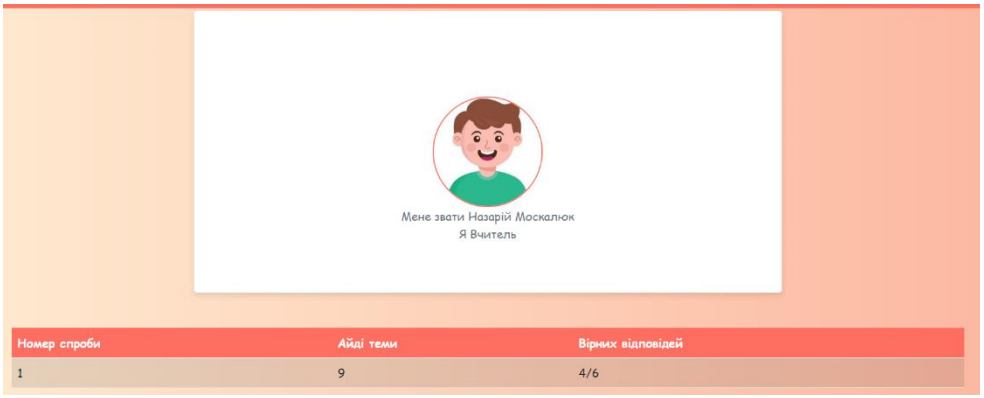

Рисунок 3.40 — Відображення результату тестування користувачем

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

4.1 Аналіз ринку збуту продукту

Проведення аналізу ринку, на якому буде проводитись збут реалізованої системи є необхідністю. Без уявлення про потенційних клієнтів та користувачів, розрахунки про витрати та доходи не можуть бути навіть близькими до реальних. Перед виставленням будь якого продукту, чи то технічного чи програмного, конкурентна спроможність грає не менш важливу роль. Інформація – ключ до успіху. Великі компанії мають свої аналогові рішення щодо навчання учнів різних вікових категорій та великий досвід в постановці та реалізації завдань.

Потенційно, веб-системи для урізноманітнення навчання, зазвичай замовляються та використовуються приватними навчальними закладами. Дуже рідко державні установи виділяють бюджет на закупівлю навчального програмного забезпечення на довгий термін, тому, найчастіше клієнтом буде приватний замовник.

Технічно, система потребує постійних витрат на хостинг серверної частини. Так як це онлайн система, і всі зміни повинні відображатись і в інших користувачів, то буде недоцільним та не розкриватиме весь функціональний потенціал сервісу.

В міру недостатніх знань для реалізації кращого та масштабнішого проекту, попит на дану веб-систему приближений до мінімальних значень. Платформа не має достатнього функціоналу, щоб замінити деякі елементи навчального процесу.

4.2 Розрахунок витрат на проектування

Етап розрахунку витрат для системи, що проектується та реалізовується для навчання учнів початкової школи повинен включати в себе витрати на проектування та розрахунок заробітної плати розробників.

Розрахунок буде проводитись використовуючи “трудовий” метод, який оформлюється в таблиці, продемонстрованій в таблиці 4.1 в Додатку Б .

Для того, щоб заповнити кошторис, необхідно провести деякі розрахунки до кожного пункту.

Для того, щоб визначити заробітну плату проектувальників веб-системи, для початку потрібно визначити кількість працівників на посаді, взяти діючі посадові оклади та час, участі в розробці. Посадовий оклад визначається за допомогою множення ставки першого розряду, що на 2024 рік становить 3600 гривень, на тарифний коефіцієнт, а так як тарифний коефіцієнт дорівнює 5,12, то оклад дорівнюватиме 3600 * 5,12 =18 432. Розрахунок відображено в таблиці 4.2.

Відрахування вираховується з урахуванням податку на доходи фізичних осіб, військового збору та єдиного внеску:

ПНДФО = 18432 * 18% = 3 317,76 грн.

ВЗ = 18432 * 1,5% = 276,48 грн.

ЄСВ = 18432 * 22% = 4 055,04 грн.

В результаті, відрахування буде 3 594,24 грн.

Таблиця 4.2 — Розрахунок заробітньої плати

N	Посада	Оклад,	Відрахуваня	Кількість		Сума
п/п	Розробник	18 432 грн/міс	3 594,24 грн/міс	1 чол.	3 місяців	14 837,76 з/п, грн.
		Усього зарплати: 44 513,28 грн.				

Соціальні потреби відраховують 22 відсотки від заробітної плати працівника, тобто відрахування = (14 837,76 * 22)/100 = 3 264,3 грн.

Інші прямі видатки складатимуть 44 відсотки від 4000 грн, тобто (4000*44)/100 = 1 760 грн.

Прямі витрати становлять 49 537,58 грн.

Накладні витрати становлять витрати для забезпечення проведення роботи та визначаються в 33% від суми прямих витрат, тобто (49 537,58 * 33%)/100 = 16 347,4 грн.

Планові накопичення вираховуються за 25 відсотками від суми накладних та прямих витрат, тобто (65 884,98 * 25%)/100 = 16 471,24 грн.

Загальна кошторисна вартість проекту рахується сумою прямих і накладних витрат додаючи до планових накопичень. 49 537,58 + 16 347,4 + 16 471,24 = 82 356,22 грн.

Податок на додану вартість рахується по 20 відсоткам від кошторисної вартості проекту, тобто (82 356,22 * 20%)/100 = 16 471,24 грн.

Договірна ціна це сума всім попередніх розрахунків. 82 356,22 + 16 471,24 = 98 827,46 грн.

4.3 Обґрунтування необхідності розробки

Потреби в системах, які будь яким способом полегшують чи заміняють той чи інший процес навчання будуть надалі продовжувати набувати популярність як і серед навчальних закладів так і серед учнів чи студентів, які можуть набути нові знання використовуючи різноманітний функціонал.

Зі зростом популярності також буде стрімкий зріст кількості та вартості реалізації схожих проектів. В більшості випадків це буде швидкість виконання, безпечне зберігання конфіденційних даних та варіативність. Найкращою тактикою буде випускати багато продуктів під одним ім’ям, проте з різним призначенням, щоб потенційний клієнт мав змогу в міру обмеженого бюджету обрати саме той продукт, який потрібно.

Досвід при роботі з клієнтами грає важливу роль. Якщо продукти, якимось чином зацікавили покупця, відгуки або статистика веб-системи стане основою

для майбутніх проектів та співпраці. При великій кількості відвідувань веб-системи клієнт може получити вигоду, до прикладу, використавши свої розробки або прорекламувавши свій бренд використовуючи внутрішню рекламу.

З точки зору учня, подібні системи неабияк сприяють розвитку. Якщо з малих років почати вивчати іноземні мови, в майбутньому буде набагато більше кваліфікованих робітників, які зможуть співпрацювати з іноземними фірмами при цьому покращуючи як своє так і державне становище, за умови того, що посада на якій працює робітник є державною.

Проектування та реалізація веб-системи яка допомагає учням вивчати іноземну мову як англійська має велику популярність та розповсюдженість не тільки в Україні. В багатьох країнах, уряд пробує розробити свою систему яка б полегшувала контроль та навчання освітніх закладів.

ВИСНОВКИ

Вебсистема для вивчення англійської мови учнями початкової школи була реалізована поетапно. Проаналізовано потенційні потреби вимоги до реалізації проекту. Важливим кроком є аналіз існуючих рішень, який допоміг зрозуміти те, який функціонал повинен бути в вебсистеми, розподіл прав доступу, які будуть впливати на можливості користування платформою шляхом реалізації входу та реєстрації.

Спроектовано макети кожної сторінки з навчальними матеріалами та тестами, розміщення елементів керування, інформаційних блоків, тематичних зображень тощо.

Було спроєктовано та розроблено базу даних, яка є невід'ємною складовою при зберіганні будь якого обсягу даних. Це важлива частина проєкту, яка дозволяє забезпечити необхідний функціонал, який в даному випадку повинен відповідати вимогам, спроектованим раніше.

Реалізовано веб-систему використовуючи новітні технології та програмні засоби, попередньо проаналізувавши можливості, які надаються розробнику. Всі тестування пройдено успішно, було охоплено більшість функціональних процесів, з якими могли виникнути ті чи інші проблеми.

Також, не варто забувати про те, що реалізація та тестування не є останнім етапом при розробці будь-якого завдання чи проєкту. В даному випадку, було також проаналізовано ринок та виконано власні розрахунки, що відображають економічну складову вебсистеми.

В подальшому буде доповнено наявний функціонал додатковими типами тестувань, та різними видами блоків, що відображатимуть необхідну інформацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Css. *MDN Web Docs*. URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата звернення: 02.05.2024).
2. Express - фреймворк веб-додатків node.js. *Express - Node.js web application framework*. URL: https://expressjs.com/ru/ (дата звернення: 30.05.2024).
3. Get started with Bootstrap. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/docs/5.2/getting-started/introduction/ (дата звернення: 05.06.2024).
4. Introduction. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (дата звернення: 06.06.2024).
5. JavaScript | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення: 23.05.2024).
6. Mdn. *MDN Web Docs*. URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення: 23.05.2024).
7. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/en (дата звернення: 26.05.2024).
8. W3Schools.com. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/sql/sql_syntax.asp (дата звернення: 02.06.2024).

ДОДАТКИ

Додаток А

Серверний файл, який містить підключення, та всі запити до бази даних.

```
const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql');

const app = express();
const port = 3000;

app.use(bodyParser.json());

app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
    res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
    if (req.method === 'OPTIONS') {
        res.sendStatus(200);
    } else {
        next();
    }
});

// Create connection to MySQL database
const db_connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'eenglish_database'
});
```

```
// Connect to MySQL database
db_connection.connect((err) => {
    if (err) {
        console.error('Error connecting to database:', err);
        return;
    }
    console.log('Connected to MySQL database');
});



app.post('/register', (req, res) => {
    const { user_name, user_surname, user_gender, user_login,
user_password, user_permition_level } = req.body;

    // Check if user_login is unique
    const checkUniqueLoginQuery = `SELECT COUNT(*) AS count
FROM user_information WHERE user_login = ?`;
    db_connection.query(checkUniqueLoginQuery, [user_login],
(err, result) => {
        if (err) {
            console.error("Error checking unique login:",
err);
            res.status(500).send('Error registering user');
            return;
        }

        const isLoginUnique = result[0].count === 0;

        if (!isLoginUnique) {
            console.error('Login is already registered!');
```

```
            res.status(400).send('Login is already
registered');



 return;
        }


        // If user_login is unique, proceed with user
registration
        const getMaxUserIdQuery = `SELECT MAX(user_id) AS
max_user_id FROM user_information`;
        db_connection.query(getMaxUserIdQuery, (err, results)
=> {
            if (err) {
                console.error("Error executing query to get
max user ID:", err);
                res.status(500).send('Error registering
user');
                return;
            }


            const user_ID = results && results.length > 0 ?
results[0].max_user_id + 1 : 1;


            const insertUserQuery = `INSERT INTO
user_information (user_id, user_name, user_surname,
user_gender, user_login, user_password, user_permition_level)
VALUES (?, ?, ?, ?, ?, ?, ?)`;
            db_connection.query(insertUserQuery, [user_ID,
user_name, user_surname, user_gender, user_login,
user_password, user_permition_level], (err, result) => {
                if (err) {
                    console.error('Error inserting user:',
err);
                    res.status(500).send('Error registering
user');
```

```
                    return;
                }
                console.log('User registered successfully');
                res.send("true");

});
        });
    });
});


app.post('/login', (req, res) => {
    const { user_login, user_password } = req.body;

    // Check if user_login is unique
    const checkUniqueLoginQuery = `SELECT COUNT(*) AS count
FROM user_information WHERE user_login = '${user_login}'`;
    db_connection.query(checkUniqueLoginQuery, (err, result)
=> {
        if (err) {
            console.error("Error checking unique login:",
err);
            res.status(500).send('Error registering user');
            return;
        }

        const isLoginUnique = result[0].count === 0;
        if (!isLoginUnique) {
            const sql2 = `SELECT user_password AS
user_password_from_db FROM user_information WHERE user_login
= '${user_login}'`;
            db_connection.query(sql2, (err1, result1) => {
                if (err1) {
                    console.error('Error:', err1);
```

```
                    res.status(500).send('Error!');
                    return;
                }
                if (result1[0].user_password_from_db ==
user_password){

let is_password_correct = true;
                    res.send(is_password_correct);
                } else {
                    console.log("Something is incorrect!");
                    is_password_correct = true;
                }
            });
        } else {
            console.error('Login is not registered!');
            res.status(400).send('Login is not registered!');
        }

    });
});

app.post("/getuserinformation", (req, res) =>{
    const { user_login } = req.body;
    const sql = `SELECT * FROM user_information WHERE
user_login = ?`;
    db_connection.query(sql, [user_login], (err, result) =>{
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });
        }
        if (result.length > 0) {
```

```
            res.json(result[0]); // Sending back the user
information
        } else {
            console.log("User not found");
            res.status(404).json({ error: 'User not found'
});
        }



});
});


app.post("/getuserinformationbyid", (req, res) =>{
    const { user_id } = req.body;
    const sql = `SELECT * FROM user_information WHERE user_id
= ?`;
    db_connection.query(sql, [user_id], (err, result) =>{
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });
        }
        if (result.length > 0) {
            res.json(result[0]); // Sending back the user
information
        } else {
            console.log("User not found");
            res.status(404).json({ error: 'User not found'
});
        }
    });
});
```

```
app.post("/getUserAttemptsToPassTopicTestInformation", (req,
res) => {
    const { user_ID } = req.body;
    const sql = `SELECT * FROM attempts_to_pass_topic_test
WHERE user_id = ?`;
    db_connection.query(sql, [user_ID], (err, result) => {
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });



}
        if (result.length > 0) {
            res.json(result);
        } else {
            console.log("Attempts not found");
            res.status(404).json({ error: 'Attempts not
found' });
        }

    });

});

app.post("/getAllTopicInformation", (req, res) => {
    const sql = `SELECT * FROM topic_information`;
    db_connection.query(sql, (err, result) => {
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });
        }
```

```
        if (result.length > 0) {
            res.json(result);
        } else {
            console.log("Topics not found");
            res.status(404).json({ error: 'Attempts not
found' });
        }

    });

});



app.post("/getInformationForTopicPage", (req, res) =>{
    const { topicId } = req.body;
    const sql = `SELECT * FROM topic_information WHERE
topic_id = ?`;
    db_connection.query(sql, [topicId], (err, result) =>{
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });
        }
        if (result.length > 0) {
            res.json(result);
        } else {
            console.log("Topics not found");
            res.status(404).json({ error: 'Attempts not
found' });
        }
    });
```

```
});


app.post("/getTopicTest", (req, res) =>{
    const { topicTestId } = req.body;


    const sql = `SELECT * FROM topic_test_information WHERE
topic_test_id = ?`;
    db_connection.query(sql, [topicTestId], (err, result) =>{
        if (err) {
            console.log(err);
            return res.status(500).json({ error: 'Internal
Server Error' });
        }


 if (result.length > 0) {
            res.json(result);
        } else {
            console.log("Test not found");
            res.status(404).json({ error: 'Test not found'
});
        }
    });


});


app.post("/sendTestAttempt", (req, res) => {
    const { user_id, topic_test_id, attempt_result } =
req.body;
    // Query to get the maximum attempt_id
    const getMaxAttemptIdQuery = `SELECT MAX(attempt_id) AS
max_attempt_id FROM attempts_to_pass_topic_test`;


    db_connection.query(getMaxAttemptIdQuery, (err, results)
=> {
```

```
        if (err) {
            console.error("Error executing query to get max
attempt ID:", err);
            res.status(500).send('Error registering user');
            return;
        }
        const attempt_id = results && results.length > 0 &&
results[0].max_attempt_id !== null ?
results[0].max_attempt_id + 1 : 1;


        const insertAttemptQuery = `INSERT INTO
attempts_to_pass_topic_test (attempt_id, user_id,
topic_test_id, attempt_result) VALUES (?, ?, ?, ?)`;
        db_connection.query(insertAttemptQuery, [attempt_id,
user_id, topic_test_id, attempt_result], (err, result) => {
            if (err) {
                console.error('Error inserting attempt:',
err);
                res.status(500).send('Error inserting
attempt');
                return;
            } else {
            console.log('Attempt inserted successfully');
            res.status(200).send('Attempt inserted
successfully');
            }
        });
    });
});


app.post("/createTopicAndTopicTest", (req, res) => {
    const { user_id, user_permition_level, topic_name,
topic_description, topic_main_information,
topic_question_one, topic_question_two, topic_question_three,
topic_question_four, topic_question_five, topic_question_six,
topic_answer_one, topic_answer_two, topic_answer_three,
```

```
topic_answer_four, topic_answer_five, topic_answer_six } =
req.body;


    const getmaxtopicid = `SELECT MAX(topic_test_id) AS
max_topic_test_id FROM topic_test_information`;
    db_connection.query(getmaxtopicid, (err, result) => {
        if (err) {
            console.error('Error getting max topic id:',
err);
            res.status(500).send('Error getting max topic
id');
            return;
        }


        const topic_id = result && result.length > 0 &&
result[0].max_topic_test_id !== null ?
result[0].max_topic_test_id + 1 : 1;


        const sql1 = `INSERT INTO topic_test_information
(topic_test_id, topic_question_one, topic_answer_one,
topic_question_two, topic_answer_two, topic_question_three,
topic_answer_three, topic_question_four, topic_answer_four,
topic_question_five, topic_answer_five, topic_question_six,
topic_answer_six) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?)`;
        db_connection.query(sql1, [topic_id,
topic_question_one, topic_answer_one, topic_question_two,
topic_answer_two, topic_question_three, topic_answer_three,
topic_question_four, topic_answer_four, topic_question_five,
topic_answer_five, topic_question_six, topic_answer_six],
(err, result) => {
            if (err) {
                console.error('Error inserting topic test:',
err);
                res.status(500).send('Error inserting topic
test');
                return;
            }
```

```
            const sql2 = `INSERT INTO topic_information
(topic_id, topic_name, topic_description,
topic_creator_information, topic_main_information,
topic_test_id) VALUES (?, ?, ?, ?, ?, ?)`;

            db_connection.query(sql2, [topic_id, topic_name,
topic_description, user_id, topic_main_information,
topic_id], (err, result2) => {

                if (err) {

                    console.error('Error inserting topic
information:', err);

                    res.status(500).send('Error inserting
topic information');

                    return;

                }


                res.status(200).send('Topic and test created
successfully');




   });

        });

    });

});


app.post("/updatesomeuserinformation", (req, res) => {

    const { user_id, userNameforchange, userSurnameforchange,
userGenderforchange } = req.body;

    const sql = `UPDATE user_information SET user_name = ?,
user_surname = ?, user_gender = ? WHERE user_id = ? `;

    db_connection.query(sql, [userNameforchange,
userSurnameforchange, userGenderforchange, user_id], (err,
result) => {

        if(err){

            console.log(err);

            return;
```

```
        } else {
            console.log("User information updated
successfully")
        }
    });
});

app.post("/gettopicpostscreatedby", (req, res) => {
    const { user_id } = req.body;
    const sql = `SELECT * FROM topic_information WHERE
topic_creator_information = ?`;
    db_connection.query(sql, [user_id], (err, result) => {
        if(err){
            console.log(err);
            return;
        } else {
            res.json(result);



 console.log("Everything's okay!");
        }
    });
});

app.post("/gettopicinformationandtestbytopicid", (req, res)
=> {
    //let { topic_id } = req.body;

    // Uncomment for testing with a fixed topic_id:
     topic_id = 2;

    const sql = `SELECT * FROM topic_information WHERE
topic_id = ?`;
```

```
    db_connection.query(sql, [topic_id], (err, result) => {
        if (err) {
            console.error("Error fetching topic
information:", err);
            res.status(500).send("An error occurred while
fetching topic information.");
        } else {
            console.log("Query result:", result);
            res.status(200).json(result);
        }
    });
});


app.post('/edittopicinformationandtest', (req, res) => {
    const {
        topic_id, topic_name, topic_description,
topic_main_information,
        topic_question_one, topic_question_two,
topic_question_three, topic_question_four,
topic_question_five, topic_question_six,
        topic_answer_one, topic_answer_two,
topic_answer_three, topic_answer_four, topic_answer_five,
topic_answer_six
    } = req.body;



    const sql = `UPDATE topic_information SET topic_name = ?,
topic_description = ?, topic_main_information = ? WHERE
topic_id = ?`;
    db_connection.query(sql, [topic_name, topic_description,
topic_main_information, topic_id], (err, result) => {
        if (err) {
            console.log(err);
            return;
        }
```

```
        const sql2 = `UPDATE topic_test_information SET
            topic_question_one = ?, topic_question_two = ?,
topic_question_three = ?, topic_question_four = ?,
topic_question_five = ?, topic_question_six = ?,
            topic_answer_one = ?, topic_answer_two = ?,
topic_answer_three = ?, topic_answer_four = ?,
topic_answer_five = ?, topic_answer_six = ?
            WHERE topic_test_id = ?`;


        db_connection.query(sql2, [
            topic_question_one, topic_question_two,
topic_question_three, topic_question_four,
topic_question_five, topic_question_six,
            topic_answer_one, topic_answer_two,
topic_answer_three, topic_answer_four, topic_answer_five,
topic_answer_six, topic_id
        ], (err, result) => {
            if (err) {
                console.log(err);
                return;
            }

 console.log('Table topic_test_information updated!');
        });
    });
});


// Start server
app.listen(port, () => {
    console.log(`Server started on port ${port}`);
});
```

Додаток Б

Таблиця 4.1 Кошторис проекту

Найменування статей витрат	Сума, грн	Обґрунтування
Зарплата проектувальників.	44 513,28	Заробітна плата, що виплачуватиметься працівникам
Відрахування на соціальні потреби.	3 264,3	Сума, що виплачуватиметься соціальним програмам та срахуванням
Контрагентські роботи і послуги.	0	
Витрати на відрядження.	0	
Інші прямі витрати.	1 760	Закупівля програмного забезпечення
Усього прямих витрат.	49 537,58	Загальна вартість прямих витрат
Накладні витрати.	16 347,4	Сума витрат пов’язані з накладними витратами
Планові накопичення.	16 471,24	Сума, виділена для реалізації майбутніх проектів
Усього, кошторисна вартість проекту.	82 356,22	Вартість проекту, що складається з прямих та накладних витрат
Податок на додану вартість.	16 471,24	Сума, яка буде сплачуватись згідно податкового законодавства
Загалом, договірна ціна розробки Зп.	98 827,46	Сума, яка відображає вартість проекту, включаючи витрати та податки