МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ ПРОВЕДЕННЯ ОПИТУВАНЬ ТА ТЕСТУВАНЬ ЗАСОБАМИ REACT, REDUX, FIREBASE»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
О.Д. Пісклов
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пісклову Олександру Дмитровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебсервісу для проведення опитувань та тестувань засобами React, Redux, Firebase

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка вебсервісу для проведення опитувань та тестувань засобами React, Redux, Firebase.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	20.02.2023	
3.	Обробка методичних та теоретичних	13.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	14.04.2023	
5.	Розробка третього розділу.	22.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	23.06.2023	

Студент _____ О.Д. Пісклов
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебсервісу для проведення опитувань та тестувань засобами React, Redux, Firebase»: 61 с., 51 рис., 4 табл., 10 джерел.

ОПИТУВАННЯ, ТЕСТИ, FIREBASE, FRONTEND, REACT, REDUX, TYPESCRIPT.

Об’єкт дослідження – процес розробки вебсервісу для проведення опитувань та тестувань за допомогою React, Redux, Firebase.

Мета роботи – аналіз предметної області та розробка вебсервісу для проведення опитувань та тестувань.

Методи дослідження – методи програмної інженерії, системний аналіз, методи функціонального програмування.

У роботі було розроблено веб-сервіс для проведення опитувань і тестувань. Також представлено огляд вже існуючих сервісів для проведення тестувань і опитувань, порівняльний аналіз цих сервісів між собою, а також порівняння цих додатків із тим, який було розроблено в роботі. Були проаналізовані вимоги до сервісу, описана його архітектура, побудована ER-діаграма бази даних і UML-діаграми. Були описані найважливіші деталі розробки, приклади коду, а також результати тестування функціоналу розробленого сервісу. Результати роботи можуть бути використані у сфері розробки веб-додатків, а також у сфері розробки додатків для проведення тестів і опитувань.

SUMMARY

Bachelor’s qualifying paper «Development of a Web Service for Conducting Surveys and Tests using React, Redux, Firebase»: 61 pages, 51 figures, 4 tables, 10 references.

SURVEYS, TESTS, FIREBASE, FRONTEND, REACT, REDUX, TYPESCRIPT.

The object of the study is the process of developing a web service for conducting surveys and tests using React, Redux, Firebase.

The aim of the study is analyze the subject area and develop a web service for conducting surveys and tests.

The methods of research are software engineering methods, system analysis, functional programming methods.

In the paper, we developed a web service for conducting surveys and tests. The paper also provides an overview of existing services for creating tests and surveys, as well as a comparative analysis of these services with each other and a comparison of these services with the application that developed in the paper. Also in the paper, the requirements for the service were analyzed, its architecture was described, and an ER diagram of the database and UML diagrams were built. In addition, the paper describes the most important details of the development, code samples, and the results of testing the functionality of the developed service. The results of work can be used in the field of web application development and in the field of developing applications for conducting tests and surveys.

ЗМІСТ

ВСТУП

Тести й опитування використовуються в багатьох сферах, починаючи від освіти, закінчуючи соціологією. Раніше їх доводилося проводити вручну, друкувати на папері, роздавати респондентам і збирати відповіді вручну. Однак із розвитком технологій зникла необхідність у цьому і зараз можна проводити їх в інтернеті. Онлайн формат проведення тестів і опитувань дає змогу дуже швидко створювати тести й опитування, розсилати їх іншим людям, а також дає змогу дуже швидко збирати результати і навіть є можливість для автоматичного створення діаграм, таблиць і графіків, які візуалізуватимуть відповіді респондентів на тест. Завдяки цьому, можна заощадити багато часу та сил.

На даний момент існує багато веб-сервісів, де можна проводити тести та опитування. Вони надають досить широкий функціонал, мають можливість для автоматичного оцінювання результатів, миттєвого редагування запитань, є можливість дивитися в реальному часі, як респондент проходить опитування, мають змогу переглянути статистику щодо відповідей респондентів, експортувати її кудись і багато іншого.

Метою кваліфікаційної роботи є створення веб-сервісу для проведення тестувань та опитувань. Для створення цього сервісу потрібно вирішити такі завдання:

– проаналізувати вже наявні аналоги веб-застосунків для проведення тестів та опитувань;

– зробити порівняльний аналіз між наявними сервісами та додатком, який розробляється;

– сформулювати вимоги до додатка;

– спроєктувати UML діаграми;

– спроєктувати ER схему бази даних;

– спроєктувати діаграму компонентів;

– протестувати функціонал додатка.

Об’єкт дослідження – процес розробки вебсервісу для проведення опитувань та тестувань за допомогою React, Redux, Firebase.

Предмет дослідження – React, Redux, Firebase.

Методи дослідження – методи програмної інженерії, системний аналіз, методи функціонального програмування.

Структурно робота складається з таких розділів: вступ, огляд застосунків тестування, проєктування, розробка та тестування, висновки, додаток, перелік посилань.

Розділ «Огляд застосунків тестування» містить аналіз вимог до розроблюваного застосунку, а також огляд таких сервісів для створення тестів і опитувань, як Google Forms, Online Test Pad, Simpoll і Survio. Крім огляду кожного з цих сервісів, у цьому розділі міститься порівняльний аналіз цих сервісів і застосунку, який розробляється в роботі. Також, у даному розділі міститься інтегральне порівняння, виконане методом Сааті, аналогів і нашого додатка. Наприкінці цього розділу міститься висновок щодо функціональності нашого сервісу, а також уже наявних аналогів у цій сфері.

Розділ «Проєктування» містить UML діаграми, на яких можна схематично побачити, який функціонал є в додатку, ролі користувачів і те, які привілеї мають різні користувачі залежно від ролі. Також, цей розділ містить ER схему бази даних та діаграму компонентів.

У розділі «Розробка та тестування» описані найважливіші деталі з розробки, а також тут містяться результати тестування функціоналу вже готового застосунку. Наприклад, у цьому розділі продемонстрований процес створення та проходження тестів і опитувань, перегляд результатів для респондента, фільтрацію і сортування тестів та опитувань на головній сторінці тощо. Також, цей розділ містить unit тестування основного функціоналу додатку, виконане за допомогою фреймворка Jest.

1 ОГЛЯД ЗАСТОСУНКІВ ТЕСТУВАННЯ

1.1 Збір та аналіз вимог

Опишемо функціональні та нефункціональні вимоги до додатку. Функціональні вимоги описуватимуть які конкретно функції повинен мати вже готовий застосунок, а нефункціональні вимоги описуватимуть, якими властивостями має володіти система, тобто це, наприклад, вимоги до безпеки даних у застосунку або вимоги до дизайну, зручності використання.

Функціональні вимоги.

У автора є можливість видаляти, редагувати та переглядати створені тести й опитування.

У користувача має бути можливість кілька разів пройти тест або опитування. Кожна спроба має фіксуватися, тобто можна подивитися, які відповіді дав користувач уперше, вдруге тощо.

Автор тесту повинен мати можливість обмежити час проходження тесту. Після того, як час закінчиться, тест має бути автоматично завершений. Крім цього, автор тесту повинен бачити, скільки часу пішло у користувача на проходження тесту.

Кожен тест та опитування повинні мати назву, анотацію (короткий опис) і категорію (математика, соціологія, політика тощо).

Автор тесту/опитування повинен мати можливість обмежити доступ до тесту і зробити так, щоб можливість пройти тест була тільки у тих, у кого є посилання (посилання розсилає автор). Або автор може зробити тест/опитування публічним, щоб будь-яка людина могла пройти його.

Якщо тест або опитування є публічним, то його може пройти будь-який користувач сайту. Такі тести й опитування розміщуватимуться на головній сторінці. Для таких тестів або опитувань має бути зручна система пошуку та фільтрації за категорією.

Тест може містити запитання з однією правильною відповіддю, з кількома правильними відповідями, завдання на відповідність, завдання на коротку відповідь [1]. Якщо це опитування, а не тест, то воно може мати ті самі типи запитань, що й тест, але ще мають бути розгорнуті відповіді та можливість завантажити файл (наприклад, фотографію).

У автора має бути можливість подивитися результати проходження тестів або опитувань, які він створив. Повинна бути можливість відповіді конкретної людини, а також щоб була можливість подивитися невелику статистику щодо відповідей всіх респондентів (наприклад, за допомогою кругової діаграми подивитися скільки відсотків обрало якусь опцію в тесті).

Автор тесту повинен мати можливість визначати кількість балів за правильні та неправильні відповіді в тестах, має бути система штрафів, за допомогою якої можна позбавити людину балів за неправильні відповіді. Особливо це актуально для завдань із кількома правильними відповідями. Наприклад, у Moodle була проблема, коли користувач міг обрати одразу всі варіанти й отримати максимальний бал за завдання.

Нефункціональні вимоги.

Інтерфейс сервісу має бути інтуїтивно зрозумілим і простим, щоб будь-який користувач міг швидко розібратися з тим, як створювати опитування або тести. Для цього потрібно уникати зайвих елементів інтерфейсу.

Сервіс має бути безпечним. Має бути присутня система реєстрації та авторизації користувачів. Створити тест можна тільки зареєстрованим користувачам. Результати тестів/опитувань може подивитися тільки та людина, яка створила тест/опитування. Видаляти та редагувати тести/опитування може тільки людина, яка їх створила, чи адміністратор.

Застосунок має бути масштабованим. Тобто якщо ми захочемо в майбутньому додати новий функціонал до нього, то архітектура застосунку має давати змогу робити це без жодних складнощів. Якщо ми, наприклад, захочемо додати новий тип запитань у тестах і опитуваннях, архітектура коду застосунку має бути влаштована так, щоб не доводилося вносити багато змін у вже наявний

код. Або, наприклад, якщо нам захочеться додати можливість змінити тему (світла, темна тощо) ми маємо спроєктувати все так, щоб процес додавання нової теми до вже існуючих був максимально простим.

1.2 Короткий огляд деяких популярних застосунків

Аналоги сервісів для проведення опитувань і тестувань.

Google Forms. Це найпопулярніший інструмент для створення опитувань і тестів, він простий, безкоштовний, підтримується Google, має інтеграцію з іншими Google сервісами, наприклад, з Google Таблицями, куди можна зберігати результати проходження тестів і опитувань. Це основні причини, чому він користується такою популярністю.

Його функціонал дуже великий. Цей інструмент дає змогу створювати опитування різних тематик, наприклад, онлайн-реєстрації на заходи, соціологічні дослідження, голосування, тести та багато іншого. Існують готові шаблони оформлення опитувань залежно від тематики. Наприклад, можна вибрати шаблон форми для реєстрації на захід і там уже буде заготовлений набір запитань, варіантів відповідей тощо, потрібно просто доповнити шаблон деякими деталями і все готово. Крім того, що тут дуже зручно створювати опитування з абсолютно різних тематик, тут також можна дивитися статистику щодо відповідей, які давали респонденти, не потрібно самому будувати діаграми, заносити відповіді до таблиці, можна одразу розпочинати аналіз результатів.

З переваг можна виділити те, що цей сервіс безкоштовний, його інтерфейс дуже зручний та інтуїтивно зрозумілий, опитування можна створити і пройти за лічені хвилини. Крім цього, є гнучкі налаштування для дизайну опитувань, тобто можна налаштувати оформлення під себе. Є дуже зручний конструктор запитань, який дає змогу додавати в опитування запитання різних типів, наприклад, запитання з одним варіантом відповіді, з кількома, з текстовим, з датою, також дає змогу додавати зображення і відео прямо в опитування.

Недоліків у цього сервісу практично немає. Є деякі обмеження, наприклад, не можна встановити обмеження за часом на опитування, що може бути корисно, наприклад, під час проведення тестувань, але ця проблема вирішується за допомогою додаткових розширень. Також у базовій версії є обмеження за типами запитань, які можна додати в опитування, але проблема теж вирішується за допомогою різних розширень.

Online Test Pad. Інструмент, який дає змогу створювати тести, опитування, кросворди, і навіть організувати систему дистанційного навчання просто всередині сервісу, де можна створювати групи учнів, давати уроки і завдання, стежити за успішністю. Але, крім освітніх тестів, тут можна створювати також особистісні та психологічні тести. Залежно від того, який тест створюється, будуть по-різному інтерпретуватися результати. Наприклад, для освітнього тесту просто визначається кількість балів за правильні відповіді, а в психологічних тестах до кожного питання може бути дана текстова розшифровка, що можуть означати відповіді респондента [2].

З переваг можна виділити те, що цей сервіс має широкий функціонал, починаючи від створення найпростіших опитувань, закінчуючи організацією системи дистанційного навчання. Крім цього, є багато типів запитань, завдяки чому можна створювати навіть інтерактивні диктанти. Також до переваг можна віднести можливість створювати сертифікати про проходження тесту, а також є можливість встановити обмеження за часом на тест, і зробити це простіше, ніж у Google Forms, де потрібно використовувати додаткові розширення для цього. Також до переваг можна віднести можливість ручної перевірки тестів і можливість додавання коментарів. Загалом, цей сервіс дуже добре підходить для освітніх цілей, для організації дистанційного навчання.

З недоліків можна виділити, що дизайн тестів застарілий, немає гнучкого налаштування оформлення для тестів і опитувань, як у Google Forms. Крім того, немає можливості зробити повний попередній перегляд тесту, який вигляд він матиме у респондентів.

Simpoll. Це сервіс для створення опитувань, голосувань і тестувань. На

відміну від попередніх двох сервісів, цей є платним. Є безкоштовна версія, але в ній багато обмежень. Наприклад, максимальна кількість респондентів – 100. Цей сервіс уже є більш професійним, ніж два попередні. Його використовують соціологи та маркетологи для проведення соціологічних опитувань, досліджень тощо [3].

Сервіс має широкий функціонал. Він дає змогу вбудувати опитування на свій сайт, як віджет, є можливість надіслати розсилку електронною поштою, також дає змогу експортувати результати опитувань в Excel. При цьому можна експортувати результати з двома видами звітів – із загальним результатом за опитуванням і за відповідями конкретного респондента. Доступна друкована версія для офлайн-опитувань. Крім того, є хороша вбудована система для аналізу результатів, яка дає змогу подивитися, як голосували чоловіки, жінки, дорослі, діти тощо. Крім цього, цей сервіс дає змогу додавати логіку в тести, наприклад, пропускати або показувати якісь запитання, якщо людина дала певну відповідь на інше запитання тощо. Є захист від повторного проходження опитування одним і тим самим респондентом. Є багато місця під файли в платних тарифах (від 250мб до 5ГБ), наприклад, під відео. Також є обмеження за терміном проведення (день, місяць тощо) і за кількістю респондентів. Доступний перегляд геолокації, а також можна подивитися, скільки часу респондент відповідав на запитання. Є можливість у реальному часі дивитися за проходженням опитування.

Основна перевага цього сервісу – це дуже широкий функціонал для створення опитувань, гнучкі налаштування для оформлення, що дають змогу налаштувати кольори, шрифти та багато іншого, а також інтуїтивно зрозумілий і простий інтерфейс для конструктора опитувань, що дає змогу створювати опитування, перетягуючи різні елементи з одного місця в інше.

З недоліків можна виділити те, що у безкоштовної версії занадто багато обмежень і присутня нав'язлива реклама.

Survio. Це ще один професійний інструмент для створення опитувань. Теж є платним, безкоштовна версія має багато обмежень, наприклад, можливість

обробити лише 100 відповідей на місяць. Цей сервіс є дуже популярним за кордоном [4].

За функціоналом він дуже схожий на попередній сервіс. Має величезну кількість готових шаблонів, типів запитань, є можливість додавати умовну логіку в опитування, додавати обмеження за часом. Можна розсилати запрошення електронною поштою, соціальними мережами, просто надіславши посилання людині. Є можливість налаштувати сповіщення про те, що хтось пройшов тест, які надходитимуть у який-небудь месенджер, на електронну пошту тощо.

Взагалі, цей сервіс майже не поступається попередньому. Оскільки обидва сервіси є платними і розраховані на професійне використання маркетологами, соціологами, вони обидва намагаються постійно впроваджувати новий функціонал і не відставати від інших платних сервісів для створення опитувань.

Переваги та недоліки у цього сервісу такі ж, як і у попереднього. Обидва мають дуже широкий функціонал, гнучкі налаштування для оформлення опитувань, але водночас у безкоштовній версії існує величезна кількість обмежень, і вона розрахована тільки для ознайомлення з основним функціоналом, а якщо людина справді захоче використовувати цей сервіс для проведення опитувань, які проходитимуть багато людей, тоді потрібно купувати платну версію.

Застосунок, який ми розробляємо. Наш застосунок помітно поступатиметься за функціоналом кожному з цих сервісів. По-перше, не буде такого гнучкого налаштування зовнішнього оформлення опитувань і тестів. На початкових етапах розробки немає сенсу фокусуватися на цьому, краще зосередити зусилля на тому, щоб усе працювало правильно, і весь необхідний функціонал був присутнім у застосунку, тобто, щоб можна було проходити тести, опитування, дивитися результати їхнього проходження. А все інше – це вже відходить на другий план і можна доробити вже потім, якщо застосунок почне набирати популярність.

З переваг нашого сервісу можна виділити те, що він буде зручним та

інтуїтивно зрозумілим, можна буде дуже легко створювати тести й опитування, а також проходити їх. Крім цього, буде багато типів запитань, які можна буде додавати до опитувань і тестів.

З недоліків можна виділити, що багато функціоналу, який є в інших сервісах, що були описані до цього, не буде присутній у нашому застосунку. Наприклад, у нашому застосунку не можна буде обирати шаблони, за якими можна було б створити опитування з різних тематик, як у Google Forms. Не буде інтеграції з іншими сервісами та додатками, наприклад, не буде системи сповіщень про те, що тест був пройдений кимось і не буде можливості експортувати результати опитувань куди-небудь.

1.3 Порівняльний аналіз існуючих сервісів та нашого застосунку

Тепер виконаємо порівняльний аналіз аналогів, які вже є в цій сфері, і сервісу, який ми будемо розробляти. Для цього складемо порівняльну таблицю (табл. 1.1) цих додатків. У цій таблиці перелічимо функціонал, який має бути у сервісів для тестів і опитувань, а потім визначимо, чи є цей функціонал у кожного із застосунків, які ми порівнюємо. Якщо функціонал присутній, у відповідному стовпчику поставимо плюс, а якщо відсутній, тоді мінус.

Таблиця 1.1 – Порівняльна таблиця існуючих сервісів та нашого сервісу

	Google Forms	Online Test Pad	Simpoll	Survio	Наш сервіс
Створення тестів	+	+	+	+	+
Створення опитувань	+	+	+	+	+
Наявність готових шаблонів оформлення	+	–	+	+	–

Продовження табл. 1.1

	Google Forms	Online Test Pad	Simpoll	Survio	Наш сервіс
Можливість дивитися статистику за відповідями	+	+	+	+	+
Можливість встановити обмеження за часом	±	+	+	+	+
Різноманітність типів запитань	+	+	+	+	–
Чи є безкоштовним	+	+	–	–	+
Інтеграція з іншими сервісами та додатками	+	–	+	+	–
Можливість експортувати відповіді респондентів у файл	+	–	+	+	–
Можливість додати умовну логіку (розгалуження)	+	–	+	+	–
Гнучкі налаштування зовнішнього оформлення	+	–	+	+	–
Можливість вставити опитування на свій сайт	+	+	+	+	–

Виконаємо інтегральний аналіз цих сервісів і нашого сервісу методом Сааті. Будемо брати певний функціонал і порівнювати його для кожного сервісу, а потім, на основі цього, визначимо інтегральний показник якості для кожного сервісу. Інтегральний показник якості – це число, яке буде загальною оцінкою певного сервісу з урахуванням того, наскільки добре реалізован кожен функціонал, який ми будемо оцінювати.

Визначимо 5 критеріїв для інтегрального порівняння. Позначимо їх, як A1, A2, A3, A4, A5.

A1 – Перегляд статистики за відповідями.

A2 – Різноманітність типів запитань.

A3 – Налаштування зовнішнього оформлення тестів і опитувань.

A4 – Експорт відповідей у файл.

A5 – Додавання умовної логіки.

Для визначення ваги кожного критерію використовується аналітична ієрархічна процедура Сааті. Складемо матрицу парних порівнянь, де вкажемо, які критерії є більш важливішими, а які – менш важливішими. Будемо проводити попарне порівняння між цими критеріями и ставити певну оцінку, ці оцінки будуть значеннями коефіцієнтів матриці парних порівнянь. Правила заповнення матриці парних порівнянь представлено у таблиці 1.2 [5].

Таблиця 1.2 – Значення коефіцієнтів матриці парних порівнянь

x_{ij}	Значення
1	і-тий критерій за важливістю такий самий як і критерій j
3	і-тий критерій дещо важливіший за критерій j
5	і-тий критерій важливіший за критерій j
7	і-тий критерій значно важливіший критерію j
9	і-тий критерій безумовно важливіший критерію j

Матриця парних порівнянь є симетричною відносно головної осі.

Наприклад, на перехресті рядку А1 та стовпця А4 маємо значення 5/1, а на перехристі рядку А4 та стовпця А1 – значення 1/5. Крім матриці парних порівнянь, визначимо також середнє геометричне та вагу для кожного критерія. Формули для обчислення цих величин буде наведено далі. Матрицю парних порівнянь, середнє геометричне та вагу критеріїв представлено у таблиці 1.3.

Таблиця 1.3 – Матриця парних порівнянь, середнє геометричне та вага критеріїв

	А1	А2	А3	А4	А5	Середнє геометричне	Вага критеріїв
А1	1	3/1	7/1	5/1	7/1	3.74	0.49
А2	1/3	1	7/1	5/1	5/1	2.26	0.29
А3	1/7	1/7	1	1/5	1/5	0.24	0.03
А4	1/5	1/5	5/1	1	3/1	0.9	0.12
А5	1/7	1/5	5/1	1/3	1	0.54	0.07

Оцінка важливості критеріїв (середнє геометричне) визначається за наступною формулою:

$$k_i = \sqrt[n]{\prod_{j=1}^{n} x_{ij}} \ ,$$

де x_{ij} – значення на перехристі і-го рядку та j-го стовпчика;

n – кількість критеріїв.

Вага критеріїв обчислюється за наступною формулою:

$$\widehat{k_i} = \frac{k_i}{\sum_{j=1}^{n} k_j},$$

На рис. 1.1 зображена діаграма вагових коефіцієнтів для критеріїв порівняння А1, А2, А3, А4, А5. На цій діаграмі можна побачити, наскільки важливим є кожен критерій, якщо порівнювати з іншими критеріями.

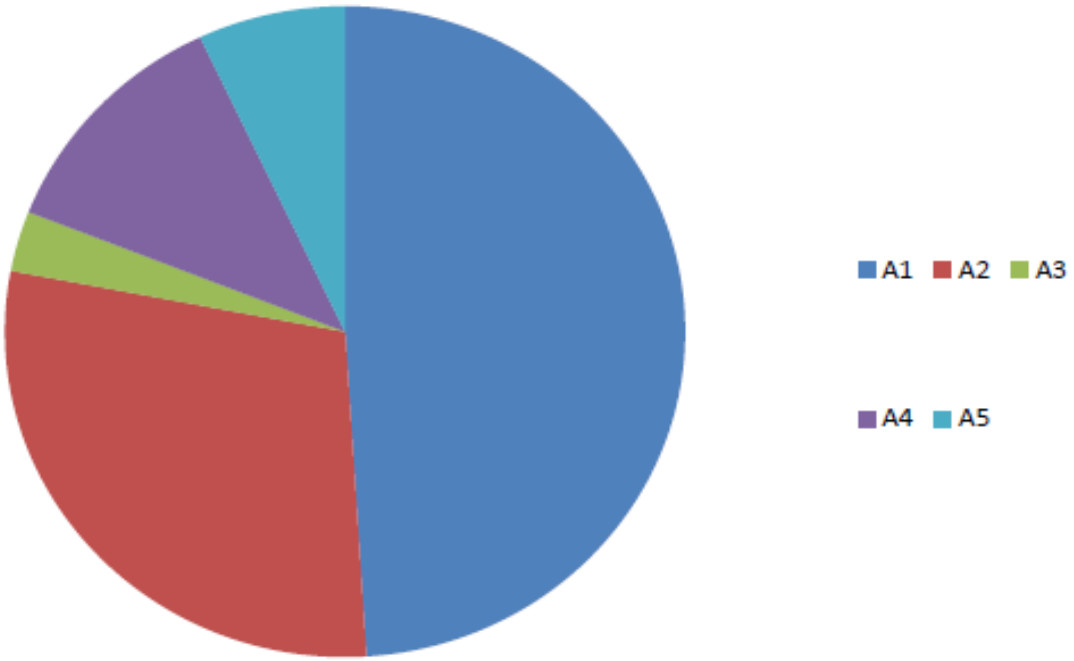


Рисунок 1.1 – Діаграма вагових коефіцієнтів

Використовуючі отримані коефіцієнти, визначимо інтегральний показник якості для програмних продуктів, що їх реалізують:

– Google Forms – ПП1;
– Online Test Pad – ПП2;
– Simpoll – ПП3;
– Survio – ПП4;
– наш сервіс – ПП5.

Будемо оцінювати одні й ті самі функції (Перегляд статистики за відповідями – А1, Різноманітність типів запитань – А2 тощо) для різних сервісів (Google Forms - ПП1, Online Test Pad - ПП2 тощо) за шкалою від 0 до 7 (0 позначає, що функція, властивість або характеристика не задовольняється, а 7 — гранично допустимий рівень їх реалізації). Після цього обчислимо інтегральний показник якості для кожного сервісу.

Формула, за якою обчислюється інтегральний показник якості:

$$Q_i = \sum_{i=1}^{n} a_i x_{ij} \ ,$$

де a_i– ваговий коефіцієнт для критерія i, x_{ij} – оцінка за шкалою від 0 до 7 критерія i для сервіса j.

Отримані результаті запишемо в таблицю 1.4.

Таблиця 1.4 – Таблиця інтегральних показників якості

Критерії	Вагові коефіцієнти	Програмні продукти					Базові значення
		ПП1	ПП2	ПП3	ПП4	ПП5	
А1	0.49	6	6	7	7	4	6
А2	0.29	5	6	7	7	4	5.8
А3	0.03	7	3	7	7	2	5.2
А4	0.12	7	3	7	7	0	4.8
А5	0.07	6	0	7	7	0	4
Інтегральні показники якості Q		5.86	5.13	7	7	3.18	5.63

На основі отриманих даних будуємо пелюсткову діаграму показників якості додатків (рис. 1.2), пелюсткову діаграму властивостей та функціональності додатків (рис. 1.3) і пелюсткову діаграму значень функціональних характеристик (рис. 1.4).

З цього порівняльного аналізу можна зробити такий висновок: наявні аналоги у сфері сервісів для проведення опитувань і тестів мають ширший функціонал, ніж той сервіс, який ми розробляємо. Особливо у випадку з платними сервісами. Це пояснюється тим, що розробкою цих сервісів займалася велика команда фахівців, вони давно вже працюють у цій сфері, а розробкою нашого сервісу займається лише одна людина.

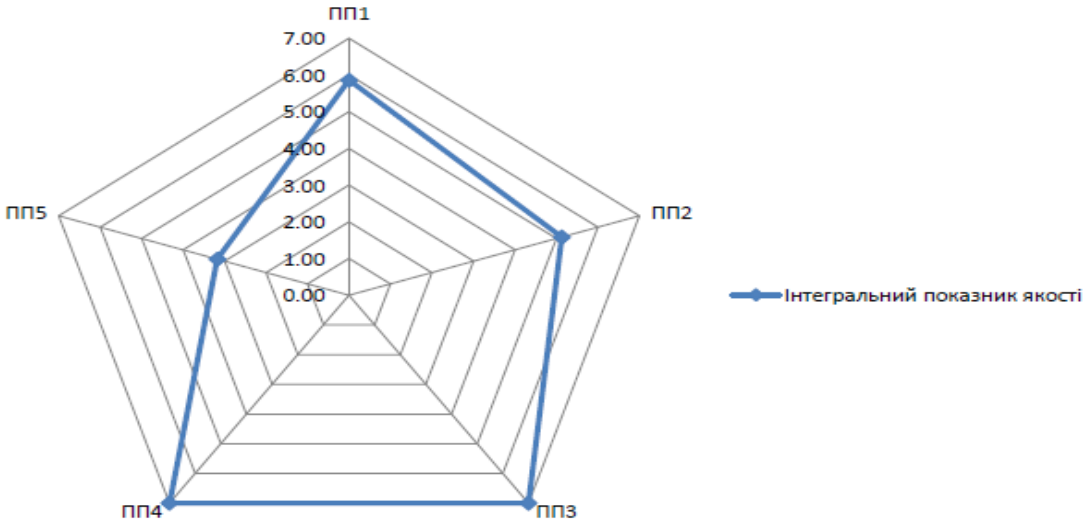


Рисунок 1.2 – Пелюсткова діаграма показників якості додатків

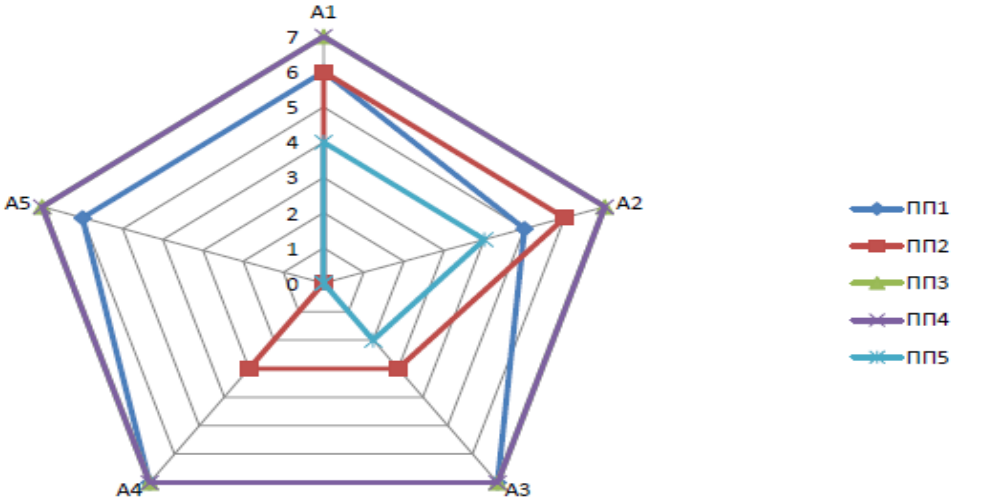


Рисунок 1.3 – Пелюсткова діаграма властивостей та функціональності додатків

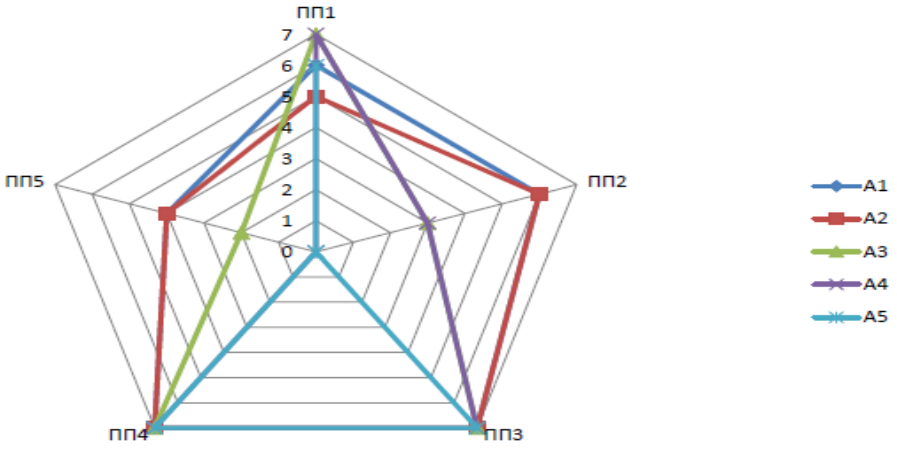


Рисунок 1.4 – Пелюсткова діаграма значень функціональних характеристик

Цілком очевидно, що застосунок або сервіс, який створено однією людиною, частіше всього, не може зрівнятися з іншими застосунками, які було створено командою професіоналів із досвідом роботи, у яких було фінансування, команда менеджерів, аналітиків тощо. Крім того, ці сервіси існують уже багато часу, їх постійно допрацьовують, виправляють якісь баги, додають новий функціонал. Якщо доопрацьовувати наш сервіс і продовжувати займатися його розробкою, можливо, він колись і зможе вийти на той самий рівень, що й наявні сервіси для проведення опитувань і тестів.

2 ПРОЄКТУВАННЯ

2.1 Побудова схем та діаграм

Побудуємо функціональну схему додатку з метою розуміння всіх функцій, які буде виконувати ПЗ. Щоб скласти функціональну систему додатка, треба розбити його на основні блоки щодо функціональності. У цьому додатку можна виділити такі основні блоки, як:

– проходження тестів та опитувань;
– створення тестів та опитувань;
– пошук, фільтрація та сортування тестів та опитувань;
– реєстрація та авторизація;
– редагування тестів та опитувань;
– видалення тестів та опитувань;
– перегляд результатів тестів та опитувань.

Функціональна схема для звичайного користувача, який проходить тести або опитування, відрізнятиметься від функціональної схеми користувача, що створює тести або опитування, тому що останній має більше можливостей для взаємодії з додатком. Щоб отримати роль автора, потрібно зареєструватися на сайті.

Тому складемо дві функціональні схеми – одна для звичайного користувача, а інша для автора тестів/опитувань. Треба зауважити, що автор тестів і опитувань володіє тими ж можливостями, що і звичайний користувач, але має додаткові привелегії. Щоб не дублювати одні й ті самі блоки, функціональна схема творця тестів/опитувань міститиме тільки додаткові можливості, які є у автора. Але взагалі, всі ті блоки, які присутні на схемі для звичайного користувача, присутні у автора теж.

Функціональна схема застосунку з точки зору автора тестів/опитувань зображена на рис. 2.1.

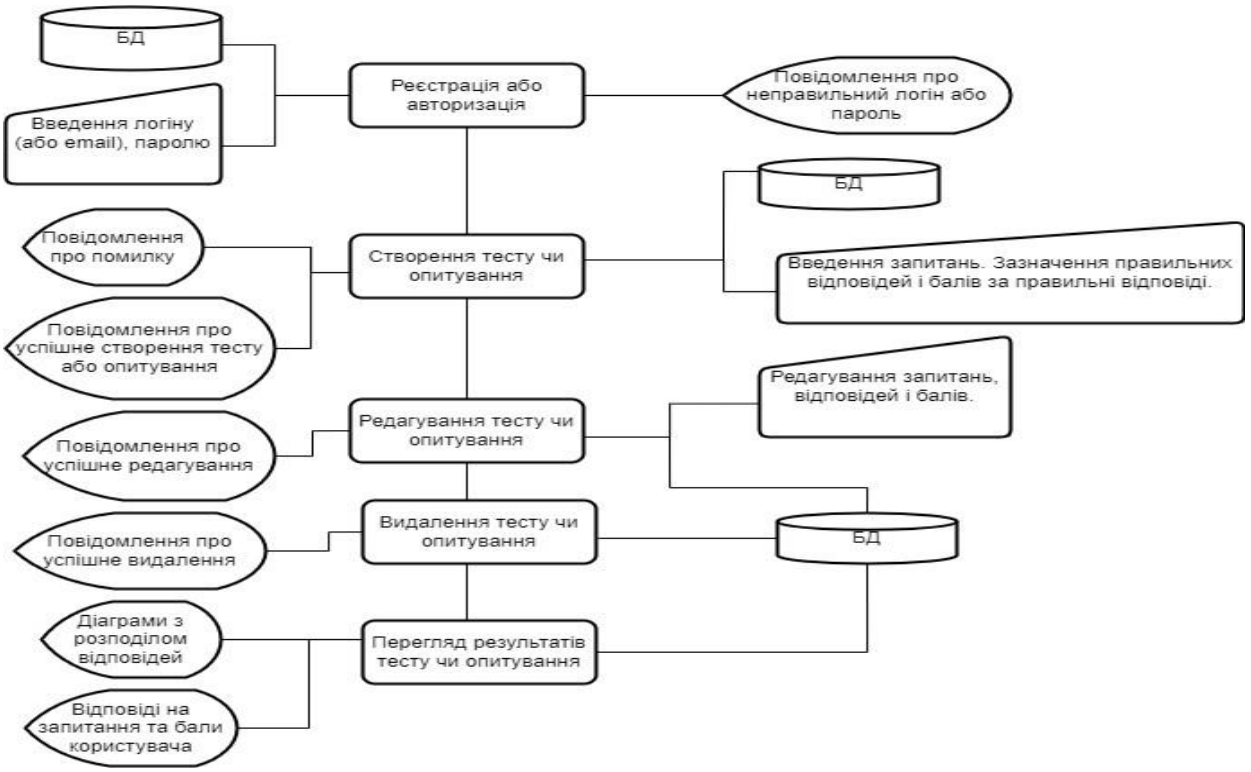


Рисунок 2.1 – Функціональна схема застосунку з точки зору автора тестів та опитувань

Функціональна схема застосунку з точки зору звичайного користувача зображена на рис. 2.2.

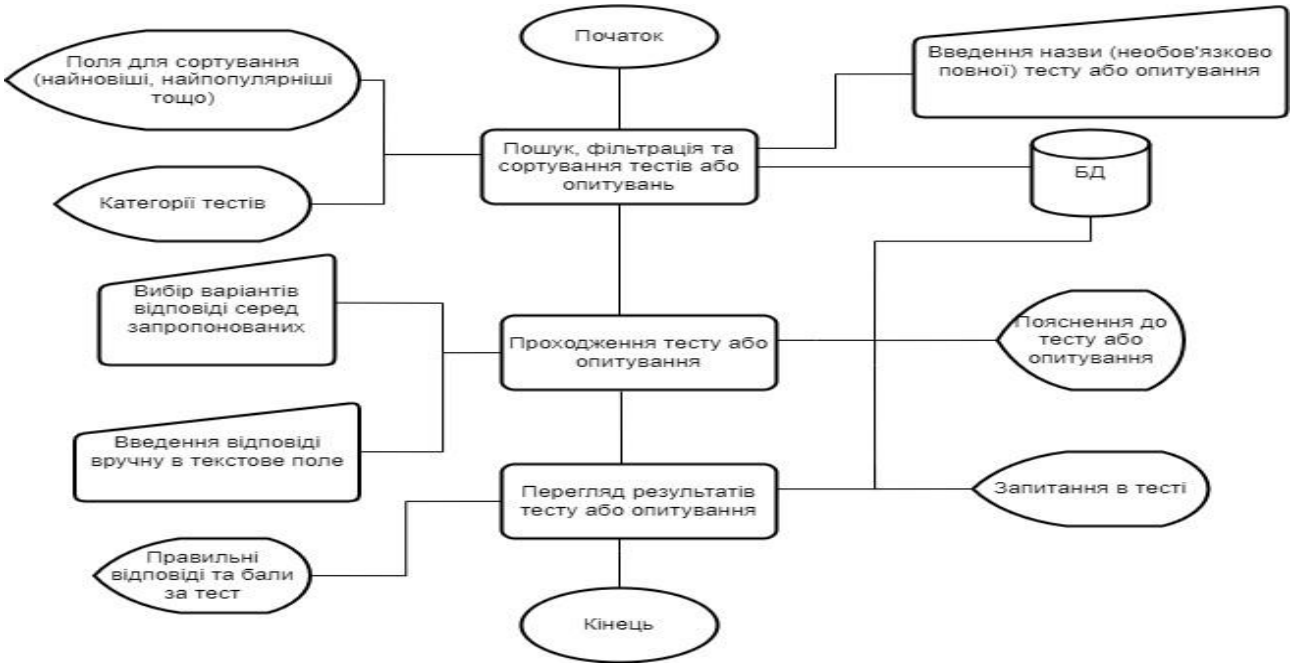


Рисунок 2.2 – Функціональна схема застосунку з точки зору звичайного користувача

Ієрархнічна (структурна) схема модулів додатку зображена на рис. 2.3.

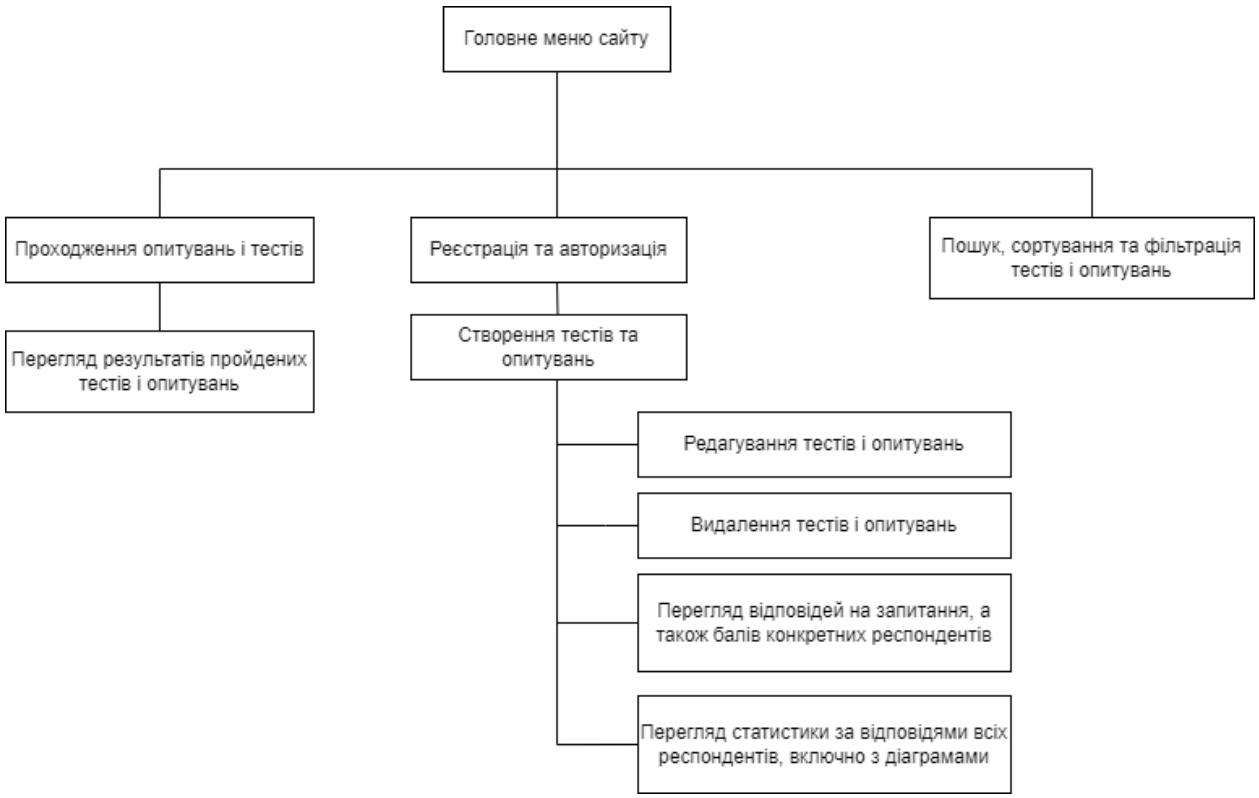


Рисунок 2.3 – Ієрархічна схема модулів

Для створення інтерфейсу сервісу буде використано бібліотеку React. Основна ідея цієї бібліотеки в розбитті користувацького інтерфейсу на складові, які називаються компонентами. Кожен компонент приймає якісь параметри, які в React називаються props (щось подібне аргументам у функціях), а потім, залежно від того, який props було передано, залежатиме, що саме буде отрисовувати компонент [6]. Крім того, у кожного компонента може бути внутрішній стан state (наприклад, якщо компонент містить інпут, то текст інпута буде внутрішнім станом state) і функції, які містять логіку. У ці функції всередині компонента зазвичай не поміщають складну логіку, тому що компонент має відповідати тільки за рендеринг певного елементу графічного інтерфейсу. Уся складна бізнес-логіка зазвичай виновиться в Redux. Завдяки цьому виконується патерн MVC – логіка додатку і зовнішній вигляд (графічний інтерфейс) розробляються окремо друг від друга.

На рис 2.4 зображено діаграму компонентів, але не для всього застосунку

(компонентів вийде надто багато і доведеться робити одразу кілька діаграм), а лише для тієї частини застосунку, яка містить функціонал проходження тестів/опитувань і перегляд результатів проходження тестів/опитувань. На діаграмі компонентів зображено компоненти, з яких складається ця частина застосунку, для кожного компонента вказано основні атрибути (props і функції), а також зображені зв’язки між компонентами.

Рисунок 2.4 – Діаграма основних компонентів

На діаграмі компонентів для атрибутів вказані типи даних із TypeScript. Деякі типи даних є елементарними. Наприклад, boolean, string і number. Але деякі типи даних не є елементарними, наприклад, Survey і Question – це користувацькі типи даних. У Typescript користувацькі типи даних створюються за допомогою інтерфейсів. В інтерфейсі ми перераховуємо, які властивості та функції мають бути у об'єктів, які успадковують інтерфейс. Деякі властивості для інтерфейсів

можна робити необов'язковими, для цього треба додати "?" після імені властивості. На рис. 2.5 зображена діаграма інтерфейсів для тієї самої частини додатка, для якої було побудовано діаграму компонентів.

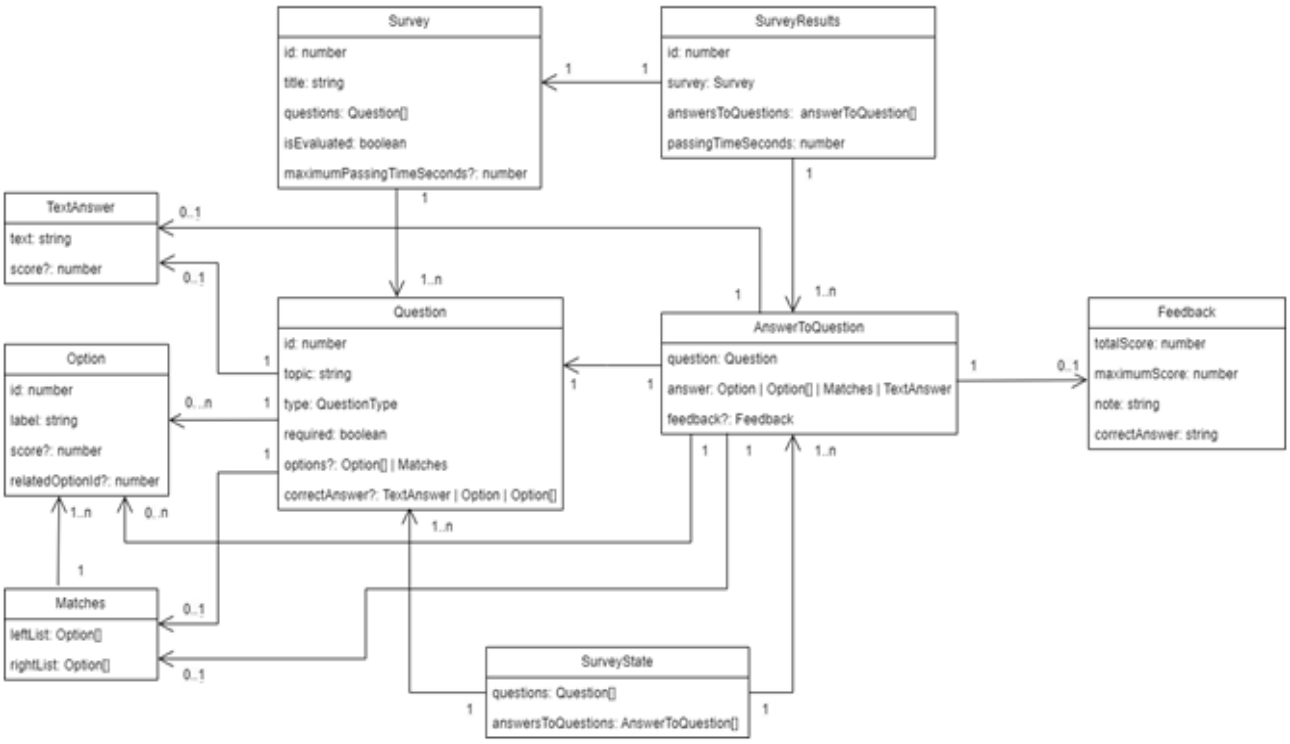


Рисунок 2.5 – Діаграма основних інтерфейсів

Цю ж діаграму ми будемо використовувати, як ER схему бази даних. Оскільки ми використовуємо Firebase Database, а ця база даних є нереляційною, вона дозволяє зберігати дані саме в такому вигляду, в якому вони зображені на схемі. Тобто, ми можемо зберігати об’єкти, які мають багато полів, всередині одного документа [7]. Наприклад, ми можемо зберігати список питань в колекції, яка містить дані про опитування. В цьому є принципова різниця реляційних та нереляційних баз даних. Якщо б ми використовували реляційну базу даних, нам потрібно було б встановлювати зв’язки між таблицями за допомогою зовнішніх ключів. В даному випадку, нам не потрібно цим займатися і це значно спрощує розробку. Це одна із причин, чому для розробки було прийнято рішення обрати саме Firebase.

Пояснення до діаграми. Користувачі проходять опитування Survey. Кожне

опитування має деякий заголовок title, при цьому є або оцінюваним (тобто тестом), або не оцінюваним. При цьому опитування може мати обмеження за часом у секундах maximumPassingTimeSeconds, після закінчення цього часу опитування автоматично завершується.

Кожне опитування Survey містить деяку кількість запитань Question. Кожне запитання Question має деяку тему topic, де написана суть запитання. Запитання можуть бути різні, тип запитання задається параметром QuestionType, який може приймати одне з таких значень:

– oneChoice (запитання, де користувач може вибрати один варіант із запропонованих);

– multipleChoice (запитання, де користувач може вибрати одразу кілька варіантів із запропонованих);

– shortTextField (запитання, де потрібна коротка відповідь);

– matchmaking (запитання на відповідності);

– detailedTextField (запитання, де треба дати розгорнуту відповідь, такі запитання можуть зустрічатися тільки в тих опитуваннях, які не оцінюються).

Запитання можуть бути як обов'язковими (required), на які користувач обов'язково має дати відповідь перед тим, як завершити опитування, так і необов'язковими. Запитання Question може мати правильну відповідь correctAnswer, якщо це тест, а може і не мати, якщо це опитування.

Запитання, де треба обирати варіанти відповіді серед запропонованих, а також запитання на відповідності містять варіанти Options. Кожен такий варіант має деякий ідентифікатор, опис label, а також може посилатися на іншу опцію із цього ж запитання (вказується ідентифікатор цієї опції), а саме в тих випадках, якщо це завдання на відповідності. Крім того, кожен варіант відповіді може мати деякий бал, якщо це тест. Якщо це правильна відповідь, то вона має бал більше нуля. Крім того, опція може мати від'ємний бал, завдяки цьому працює система штрафів, коли у користувача віднімають бали за неправильні відповіді в тесті.

Запитання на відповідності містять Matches – це два списки з варіантами. Потрібно вибрати правильну відповідність для варіантів із лівого списку, у

правому списку.

Після проходження тесту створюється об'єкт SurveyResults, куди поміщається сам Survey, де міститься вся інформація про опитування, яке було пройдено. Також у цьому об'єкті містяться відповіді, які дав користувач answersToQuestions, а також час (у секундах) passingTimeSeconds, за який користувач пройшов опитування.

Об'єкт answerToQuestion містить у собі саме запитання question, на яке користувач давав відповідь, відповідь answer, яка може бути: одним варіантом Option, кількома варіантами Option, набіром відповідностей Matches або текстовою відповіддю TextAnswer.

Відповідь на запитання може мати відгук feedback (у тих випадках, коли це тест), де вказується максимально можлива кількість балів maximumScore за запитання та скільки балів набрав користувач totalScore. У відгуку є зауваження note, де вказується, правильну відповідь дав користувач чи ні. Якщо відповідь неправильна, то користувачеві показують правильну відповідь correctAnswer (або правильні відповіді, якщо їх кілька).

На рис. 2.6 зображена діаграма послідовності, яка описує той самий функціонал системи, про який ішлося до цього, з точки зору поведінки системи.

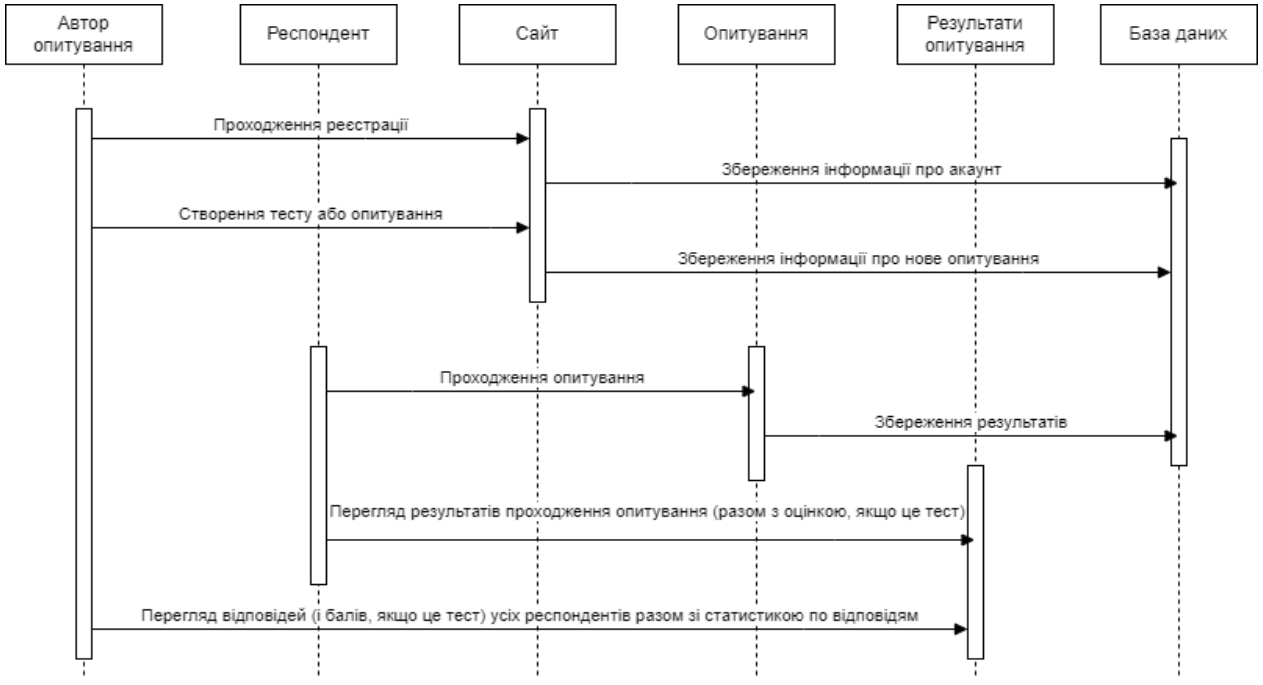


Рисунок 2.6 – Діаграма послідовності

2.2 Вибір та обґрунтування архітектурних патернів

Для розробки цього сервісу застосовуються різні архітектурні патерни, завдяки яким буде простіше вносити зміни до коду в майбутньому, а також для читабельності коду, щоб інший розробник, якщо він долучиться до розробки, міг швидко розібратися в тому коді, що вже був написаний.

Для розробки цього сервісу використовується Redux. Redux є бібліотекою і патерном одночасно. Redux дає змогу ізолювати всю логіку застосунку в окремий модуль. Можна провести аналогію з відомим архітектурним патерном MVC, згідно з яким увесь застосунок ділиться на три окремі частини – дані, графічний інтерфейс та обробники подій. Основна мета цього підходу – розділити бізнес-логіку застосунку та графічний інтерфейс. В нашому випадку, React частина застосунку міститить ту частину, яка відповідає за графічний інтерфейс застосунку, а вся основна логіка буде ізольована в Redux [8]. Якщо треба буде щось змінити в логіці застосунку, ми не будемо лізти в код React, для цього нам треба буде робити зміни в коді Redux. При цьому, ми можемо додавати нові фічі для графічного інтерфейсу в React, не змінюючи код Redux.

Центральним поняттям у Redux є store (сховище). У ньому міститься глобальний стан (state) застосунку. Глобальний стан – це такі дані, до яких ми можемо звернутися з будь-якого компонента в React частині. Взагалі, у React-компонентів може бути і свій стан, який вони можуть передавати іншим компонентам, змінювати його тощо, але оскільки React-частина застосунку не повинна містити багато логіки, цей стан переносять у Redux.

Взаємодія зі сховищем store не відбувається безпосередньо. Щоб взаємодіяти зі сховищем використовуються об'єкти, які називаються actions. Action описує, що в додатку відбулася якась подія. Щоб викликати Action, використовується Dispatch.

Після того, як ми викликали Action за допомогою Dispatch, Store запускає Reducer, який бере старий стан і оновлює його на новий. Після цього користувацький інтерфейс застосунку отримує повідомлення про те, що стан

застосунку змінився, потім відбувається перерендерінг компонентів з урахуванням зміненого стану [9].

Наприклад, якщо нам потрібно збільшить лічильник на одиницю після натискання на кнопку, то цей процес відбуватиметься так:

- користувач натискає на кнопку;
- за допомогою dispatch викликаємо action;
- сховище store, яке зберігає стан лічильника, викликає функцію reducer, ця функція бере старий стан лічильника, змінює його на новий (тобто додає одиницю до попередього значення);
- інтерфейс програми отримує повідомлення про те, що в сховищі store щось змінилося і після цього компонент, де міститься лічильник, перемальовується і лічильник збільшується на одиницю.

На рис. 2.7 зображена схема, на якій можна побачити, як пов’язані основні частини, з яких складається додаток, який використовує архітектуру Redux.

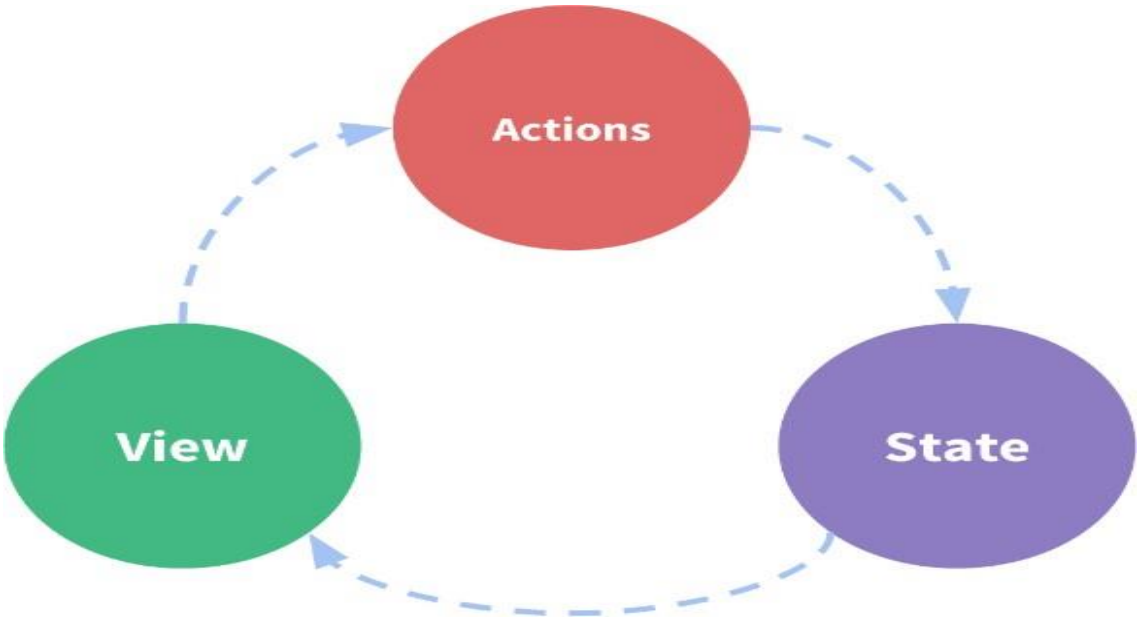


Рисунок 2.7 – Схема основних частин, з яких складається додаток з архітектурою Redux

3 РОЗРОБКА ТА ТЕСТУВАННЯ

3.1 Розробка та приклади програмної реалізації

Коли користувач заходить на сайт, він потрапляє в головне меню, де він може проходити опитування або тести, які є в списку. У головному меню також розташований пошуковий рядок і різні кнопки для фільтрації (рис. 3.1).

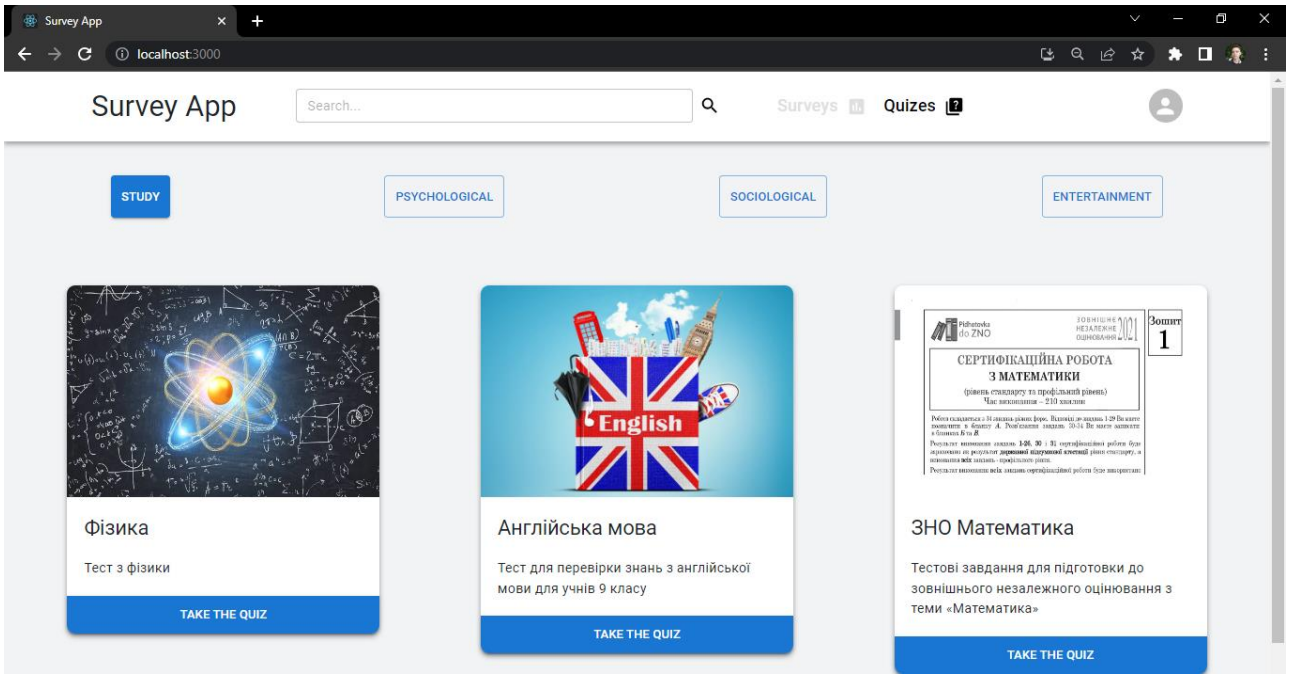


Рисунок 3.1 – Головне меню сайту

Оскільки цей сервіс дає змогу проходити як опитування, так і тести, є кнопки, які дають змогу перемикатися між опитуваннями і тестами. Вони розташовані трохи правіше від пошуку. Активна кнопка, яку вибрав користувач, підсвічується чорним, а неактивна – сірим. У пошуковий рядок користувач може ввести якесь слово або навіть частину слова, і додаток автоматично знайде збіги за заголовком і описом тесту або опитування. Якщо конкретніше, пошук враховує будь-які входження того рядка, який ввів користувач. Тобто пошук враховує, що рядок, який ввів користувач, може знаходитися на початку, в середині або наприкінці заголовка або опису.

На рис. 3.2 зображений приклад роботи пошукової системи сайту. Тут ми ввели фразу, яка є кінцем опису одного з тестів, що був на сайті, і сайт автоматично прибрав зі списку всі варіанти, крім того, який нам потрібен. При цьому не обов'язково натискати кудись, щоб підтвердити пошуковий запит, достатньо просто набрати якийсь текст у рядок для пошуку, і список опитувань автоматично змінюватиметься з урахуванням пошукового запиту.

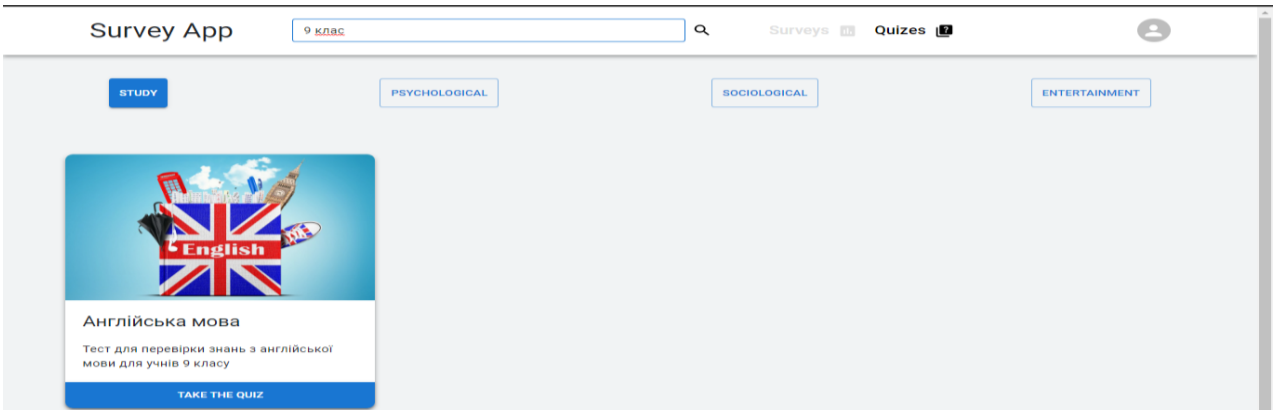


Рисунок 3.2 – Приклад роботи пошукової системи сайту

Також у шапці сайту є іконка з аватаром, натиснувши на яку, користувач може створити новий тест або опитування, а також видалити або редагувати вже створені ним тести й опитування, а також дивитися результати проходження своїх тестів і опитувань іншими користувачами.

В основній частині сайту (рис. 3.3) розташовані картки з опитуваннями або тестами (залежить від того, який фільтр встановив користувач).

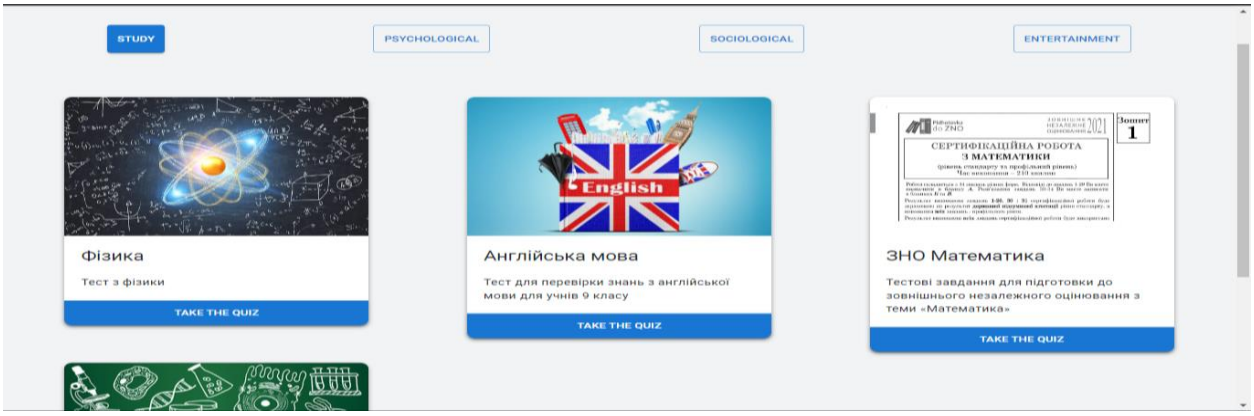


Рисунок 3.3 – Основна секція сайту

На кожній картці зображена якась картинка, посилання (URL) на яке вказує автор тесту або опитування при створенні, нижче розміщено заголовок і опис, а в самому низу картки – кнопка, натиснувши на яку, користувач перейде на сторінку, де він буде проходити тест або опитування. У верхній частині цієї секції сайту знаходяться кнопки з категоріями. Кожен тест або опитування має деяку категорію, яка вказується при створенні, і на сайті є можливість фільтрувати тести й опитування з урахуванням цієї категорії.

Щоб обрати категорію, треба натиснути на кнопку з відповідним написом. Після цього натиснута кнопка змінить свій колір і стане активною, а решта будуть неактивні. На рис. 3.4 зображений приклад роботи фільтра тестів на сайті.

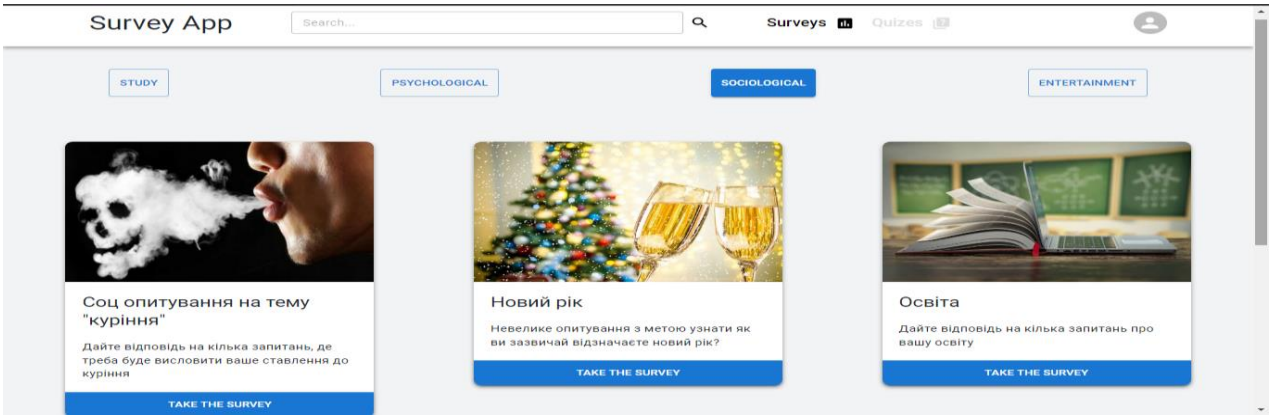


Рисунок 3.4 – Приклад роботи фільтру тестів

Якщо виникне ситуація, що в базі немає опитувань або тестів за тими критеріями, які вказав користувач, користувач отримає повідомлення про це (рис. 3.5).

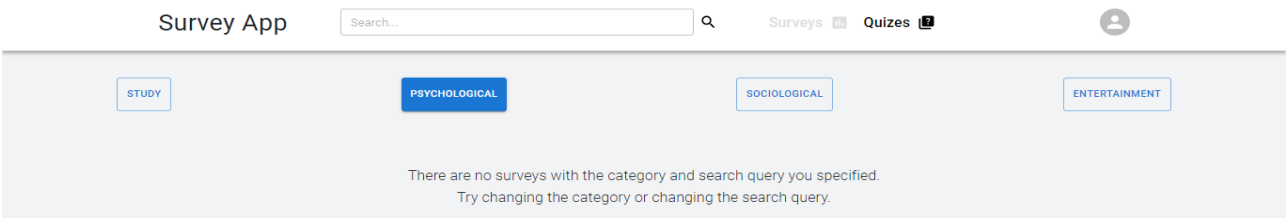


Рисунок 3.5 – Повідомлення про те, що на сайті немає опитувань, які задовольняють пошуковим критеріям

Якщо користувач натисне на кнопку «Take the quiz», відбудеться автоматичний перехід на сторінку з тестом або опитуванням. Ця сторінка складається з декількох секцій. Верхня секція містить заголовок і опис опитування, а також картинку (рис. 3.6).

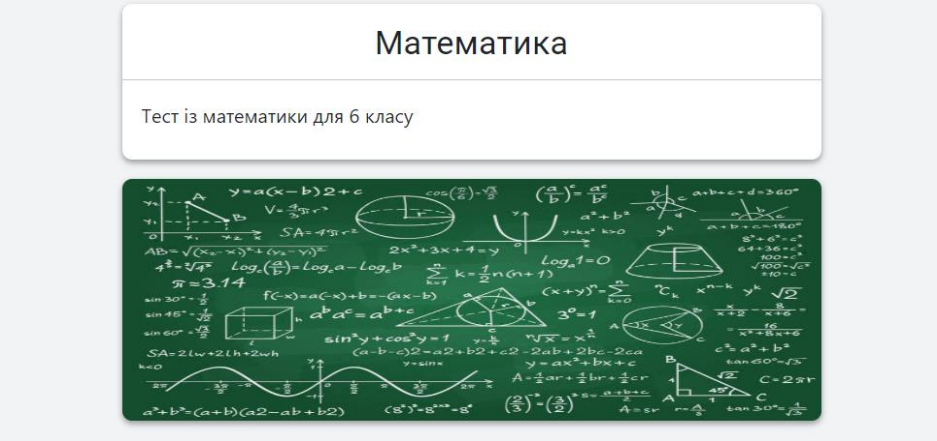


Рисунок 3.6 – Верхня секція сторінки проходження тесту або опитування

Далі, в основній секції, йдуть запитання, на які має відповісти користувач (рис. 3.7). При цьому деякі запитання є обов'язковими, і, якщо користувач не дасть на них відповідь, він не зможе завершити тест або опитування, за винятком тих випадків, якщо воно має обмеження за часом. У цьому випадку, тест або опитування буде автоматично завершено після закінчення цього періоду часу, а обов'язкові запитання можуть залишитися без відповіді.

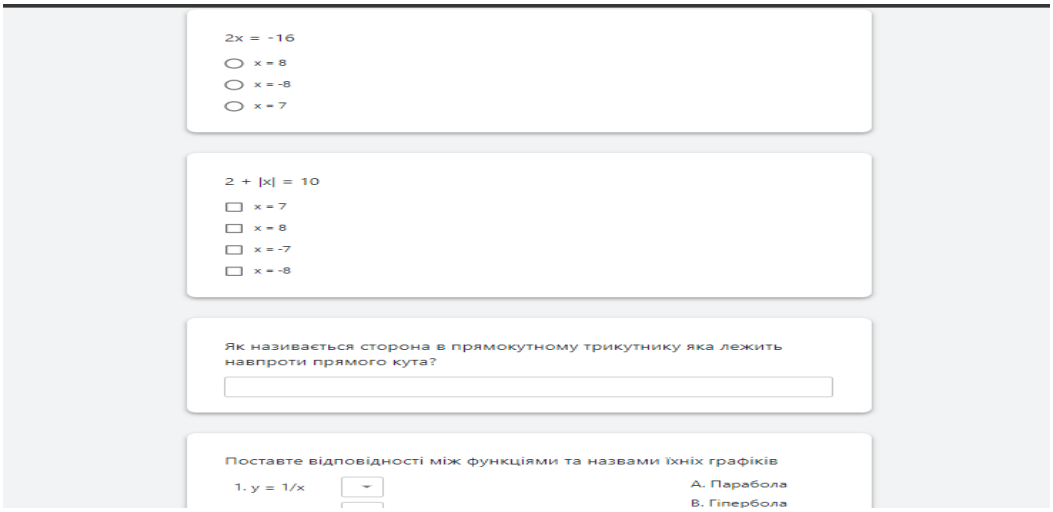


Рисунок 3.7 – Основна секція сторінки проходження тесту або опитування

Внизу сторінки знаходиться кнопка, натиснувши на яку, можна завершити тест. Якщо користувач не відповість хоча б на одне обов'язкове запитання і натисне на цю кнопку, він отримає відповідне сповіщення (рис. 3.8). Це повідомлення зникне саме через кілька секунд або користувач сам може закрити його.

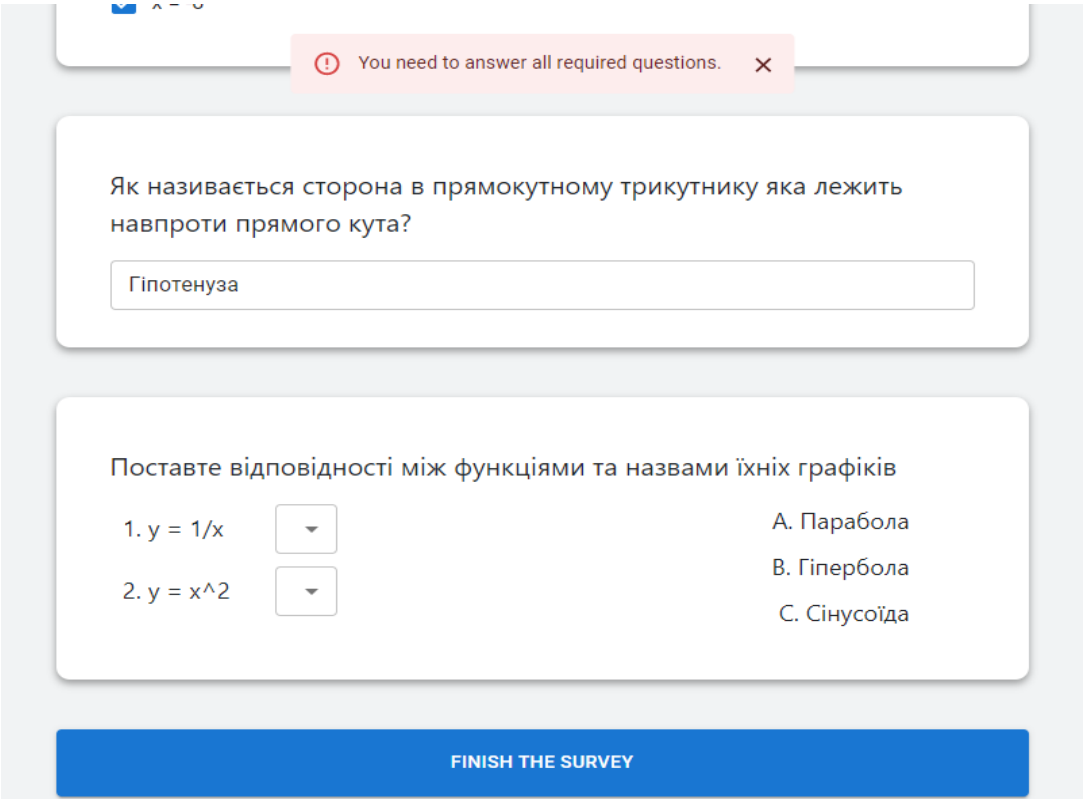


Рисунок 3.8 – Сповіщення про те, що потрібно відповісти на всі обов’язкові запитання

Якщо користувач відповість на всі запитання і збереження відповідей до бази даних пройшло успішно (якщо не виникло жодних проблем, наприклад, з інтернет-з'єднанням у користувача), тоді він отримає повідомлення про те, що всі його відповіді було збережено (рис. 3.9), а також, побачить вікно, яке перекриватиме весь інший контент сторінки, там буде зазначено оцінку (якщо це тест, а не опитування), а також тут будуть дві кнопки. Натиснувши на першу кнопку, користувач може перейти на сторінку з результатами тесту, а натиснувши на другу – повернутися на головну сторінку сайту.

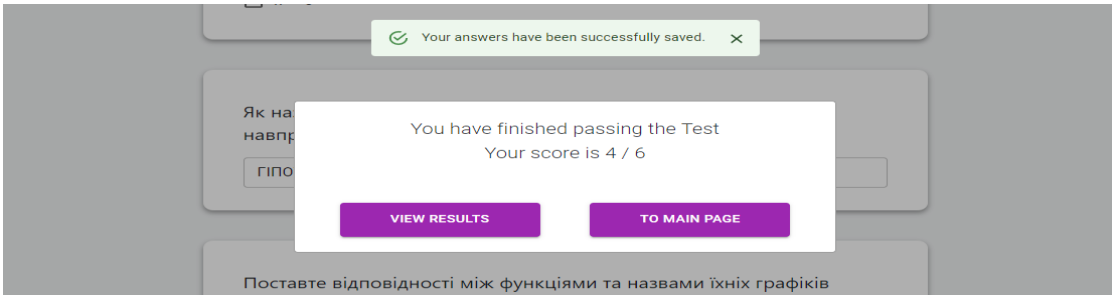


Рисунок 3.9 – Сповіщення про успішне збереження відповідей

Якщо користувач не встигне пройти тест за відведений час, тест буде автоматично завершено, підрахунок результатів враховуватиме тільки ті відповіді, які користувач встиг дати за цей проміжок часу. А також користувач отримає спливаюче повідомлення про те, що час закінчився (рис. 3.10).

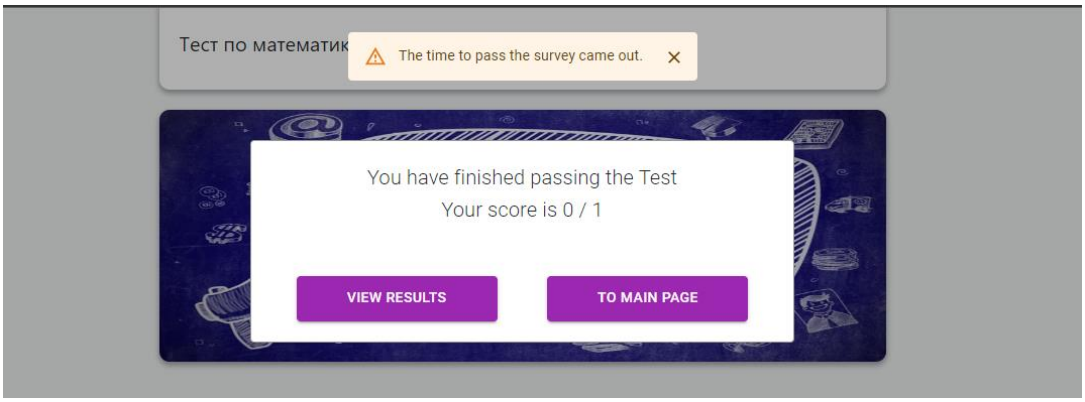


Рисунок 3.10 – Сповіщення про те, що час на проходження закінчився

Сторінка з результатами за своєю структурою нагадує сторінку, де користувач проходив тест або опитування. У верхній частині теж міститься опис тесту чи опитування, його картинка, а далі йдуть запитання, на які відповідав користувач, але окрім самих запитань, можна побачити відповіді, які дав респондент разом з відгуком (якщо це тест), а в самому низу міститься кнопка, натиснувши на яку, користувач може перейти на головну сторінку сайту.

Якщо респондент проходив тест, то відповіді респондента автоматично оцінюються. Якщо відповідь правильна, ставиться галочка, а якщо неправильна – хрестик. Крім цього, під запитанням можна побачити коментар. Є три типи коментарів: відповідь правильна, відповідь частково правильна і відповідь

неправильна. Коментар завжди містить оцінку, скільки респондент набрав балів за конкретне запитання, а також максимальну кількість балів, яку можна було заробити.

Якщо відповідь правильна, тоді коментар буде зеленого кольору і в коментарі буде вказано, що респондент набрав максимальну кількість балів за запитання. Крім цього, стоятиме галочка біля обраного варіанту (рис. 3.11).

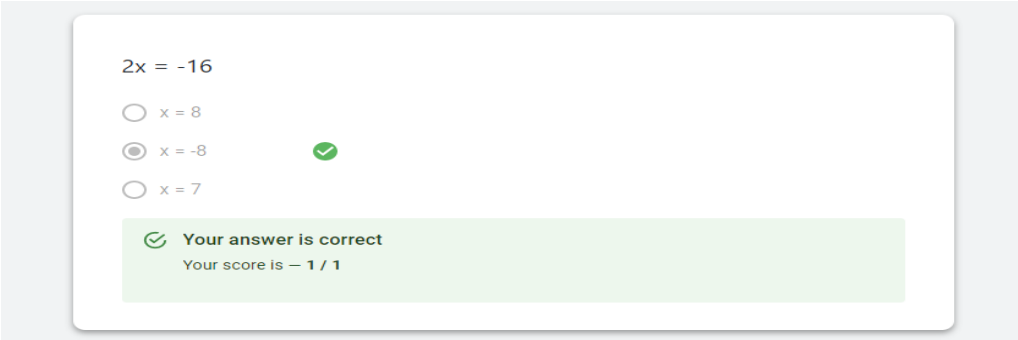


Рисунок 3.11 – Правильна відповідь в тесті

Якщо відповідь частково правильна, коментар буде жовтого кольору і в коментарі крім кількості балів, які користувач заробив, будуть вказані правильні відповіді на запитання (рис. 3.12).

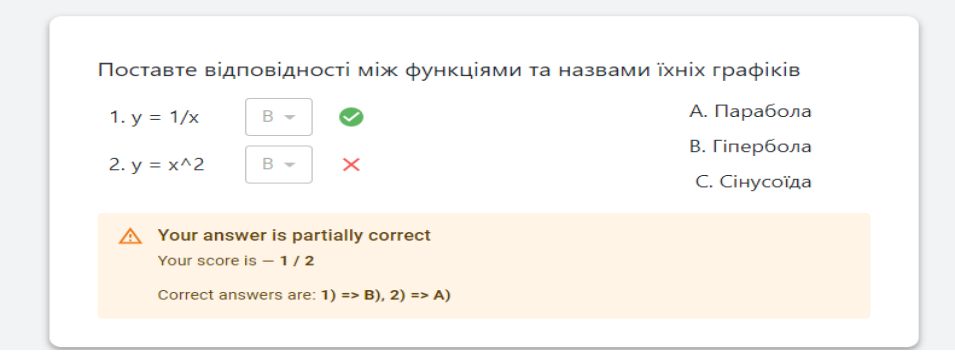


Рисунок 3.12 – Частково правильна відповідь

Якщо відповідь неправильна, коментар буде червоного кольору і будуть вказані правильні відповіді, як у випадку з частково правильною відповіддю. При цьому, якщо це текстова відповідь, а не варіант відповіді зі списку, тоді замість галочки або хрестика змінюватиметься колір межі поля, куди користувач вводив

відповідь. Якщо відповідь правильна, межа буде зеленою, а якщо неправильна – червоною (рис. 3.13).

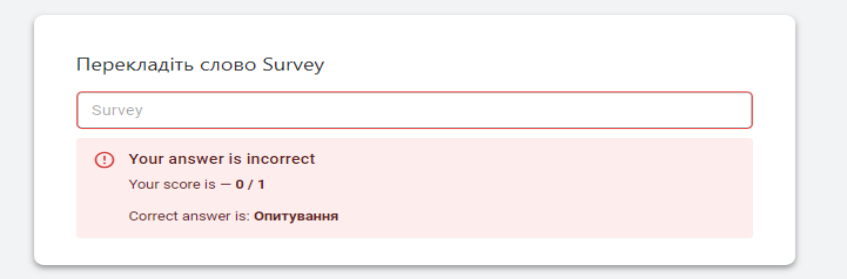


Рисунок 3.13 – Неправильна відповідь

Для створення тесту або опитування на головній сторінці потрібно натиснути на іконку з аватаром, вибрати із випадаючого списку «Create a survey», і після цього відбудеться перехід на сторінку створення опитувань. Спочатку користувач побачить форму, де потрібно вказати загальну інформацію про опитування: назву, опис, категорію, посилання на якусь картинку для опитування, чи є обмеження за часом, якщо є, то яке конкретно, а також чи буде опитування оцінюваним, тобто чи буде це тест (рис. 3. 14).

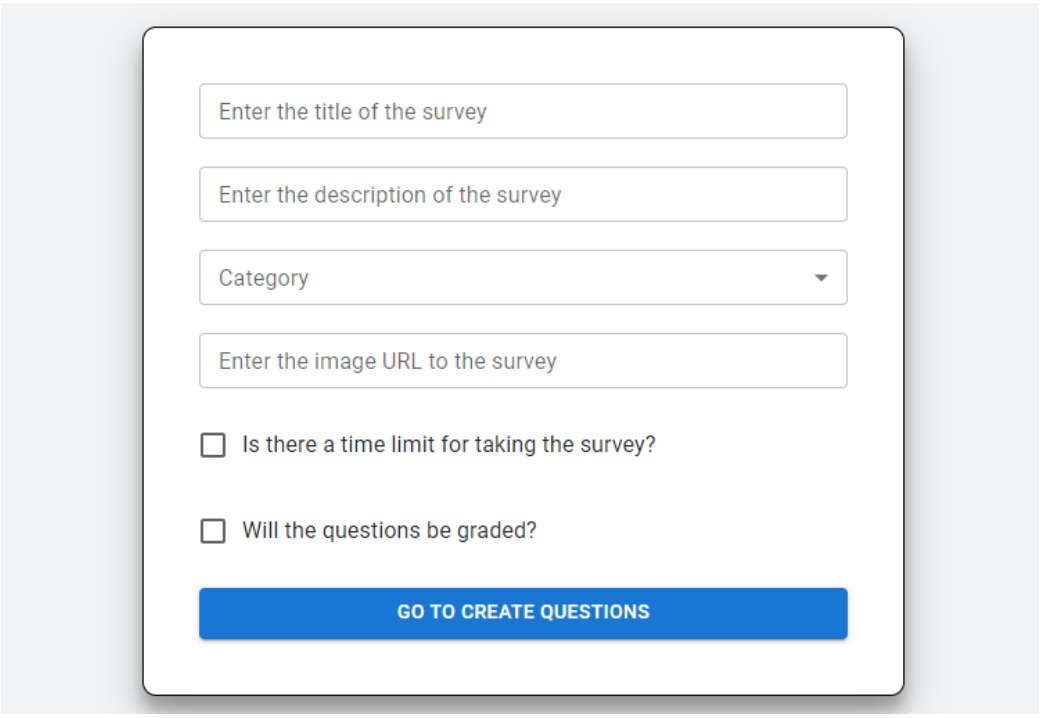
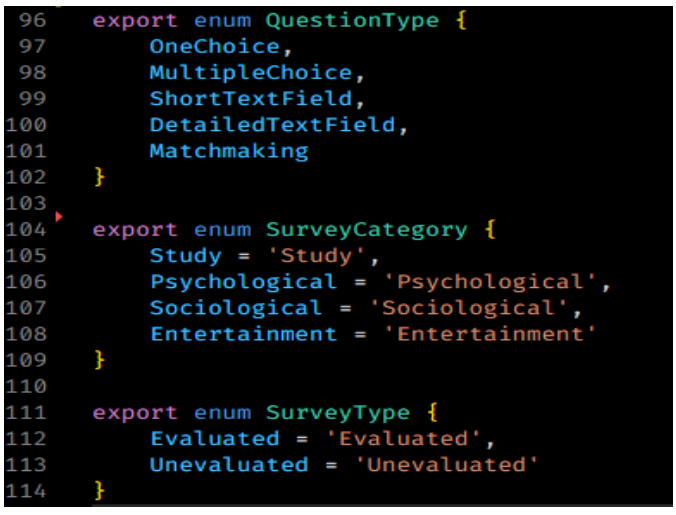


Рисунок 3.14 – Початкова форма створення тестів та опитувань

Категорія для опитування вибирається зі списку, не можна вписати якусь свою категорію. Це було зроблено для того, щоб користувачі не змогли створити величезну кількість різних категорій, чим менше категорій, тим простіше буде фільтрувати опитування за цими категоріями. Якщо дозволити користувачам вказувати категорію самостійно, це призведе до того, що на головній сторінці буде величезна кількість різних кнопок з категоріями і буде неможливо знайти потрібну категорію. Є чотири категорії опитувань: навчальні, соціологічні, психологічні та розважальні. Якщо потрібно буде додати нову категорію, це можна буде без проблем зробити, внісши при цьому мінімальні зміни в код.

У Typescript є структура даних, яка називається Enum (перерахування), тут можна задати список якихось констант, які потім будуть використані в іншому місці. У проєкті є окремий файл types/survey.ts, де описуються різні типи, інтерфейси та перерахування. Щоб додати нову категорію для опитувань, достатньо буде перейти до цього файлу і додати в перерахування SurveyCategory новий запис, який відповідатиме новій категорії. Крім цього, тут же можна додати новий тип запитань і багато іншого. На рис. 3.15 зображено фрагмент коду з файлу types/survey.ts.



```
96   export enum QuestionType {
97       OneChoice,
98       MultipleChoice,
99       ShortTextField,
100      DetailedTextField,
101      Matchmaking
102  }
103
104  export enum SurveyCategory {
105      Study = 'Study',
106      Psychological = 'Psychological',
107      Sociological = 'Sociological',
108      Entertainment = 'Entertainment'
109  }
110
111  export enum SurveyType {
112      Evaluated = 'Evaluated',
113      Unevaluated = 'Unevaluated'
114  }
```

Рисунок 3.15 – Фрагмент коду з файлу types/survey.ts

Якщо не заповнити якісь поля у формі (рис. 3.14) або заповнити їх неправильно (наприклад, вказати менше символів, ніж мінімальна кількість), тоді

під тим полем, де була допущена помилка, з'явиться повідомлення про те, що була зроблена помилка під час заповнення поля, а також вказується, що саме це за помилка, а текстове поле, де користувач ввів щось неправильно, підсвічується червоним (рис. 3.16).

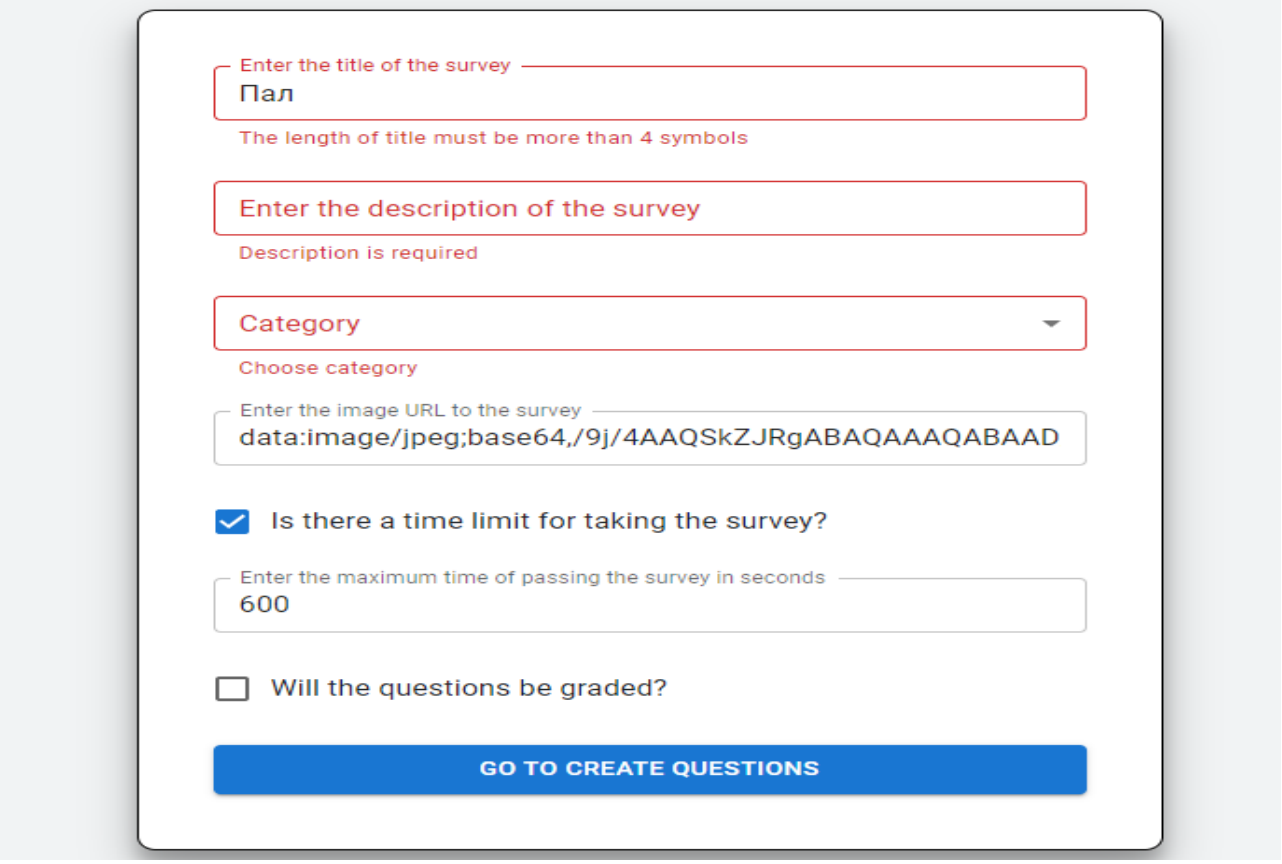


Рисунок 3.16 – Помилки при заповненні форми

Для створення цієї форми було використано популярну бібліотеку React-hook-form, вона помітно спрощує роботу з формами, за допомогою цієї бібліотеки дуже зручно робити валідацію форм, можна писати набагато менше коду, ніж якщо вручну робити валідацію. Крім того, що користувач не може ввести неправильні дані у форму перед тим, як продовжити створення опитування або тесту, він також бачить, що конкретно він ввів неправильно, а це робить процес створення опитувань простим та прозорим. Помилки прибираються відразу в той момент, коли валідація пройдена успішно, користувачеві не треба багато разів натискати на кнопку «Go to create questions». Фрагмент коду, який відповідає за цю форму, представлено на рис. 3.17.

```
<TextField
    label = "Enter the title of the survey"
    size = 'small'
    fullWidth
    {
        ...register(
            'title', {
                required: 'Title is required',
                minLength: {
                    value: parseInt(process.env.REACT_APP_MIN_TITLE_LENGTH || '4'),
                    message: `The length of title must be more than
                    ${process.env.REACT_APP_MIN_TITLE_LENGTH} symbols`
                }
            }
        )
    }
    sx = {{ marginBottom: '20px' }}
    error = {errors.title?.message !== undefined}
    helperText = {errors.title?.message}
/>
```

Рисунок 3.17 – Фрагмент коду форми створення тестів та опитувань

На рис. 3.17 можна помітити, що для створення текстового поля використовуються не звичайний html тег input, а спеціальний React компонент. Цей компонент створювався не вручну. Для створення різних кнопок, текстових полів, списків та інших елементів графічного інтерфейсу було використано бібліотеку React MUI. Ця бібліотека надає величезну кількість уже готових компонентів, з яких можна створювати користувацький інтерфейс. Для того щоб змінювати зовнішній вигляд цих елементів, не потрібно писати CSS, а досить просто змінювати атрибути React компонента, наприклад, можна вказати розмір текстового поля, задавши значення атрибуту size або додати підказку для введення, якщо користувач ввів щось неправильне, за допомогою атрибута helperText [10].

Мінімальна кількість символів для кожного поля задається, як змінна оточення. Усі ці змінні знаходяться в одному файлі і, якщо потрібно буде змінити якісь мінімальні значення довжини для якогось поля, не потрібно шукати ці змінні в коді, який відповідає за форму, достатньо буде просто перейти до файлу зі змінними оточення .env. Крім цього, оскільки користувач може ввести неправильний URL картинки, має бути якась стандартна картинка для опитувань і тестів, щоб не зіпсувався дизайн сайту, коли одні картки з опитуваннями мають картинку, а інші ні. У цьому ж файлі .env задано посилання на картинку, яка буде встановлена за замовчуванням, якщо користувач вкаже неправильний URL. Вміст файлу .env продемонстрований на рис. 3.18.

```
.env
1  REACT_APP_MIN_TITLE_LENGTH = 4
2  REACT_APP_MIN_DESCRIPTION_LENGTH = 4
3  REACT_APP_MIN_IMAGE_URL_LENGTH = 6
4  REACT_APP_DEFAULT_SURVEY_IMAGE_URL = https://st3.depositphotos.com/1092019/12572/i/450/depositphotos_12
```

Рисунок 3.18 – Вміст файлу .env

Тут же можна буде задати максимальну довжину для заголовка, опису, URL та інших параметрів, якщо вони з'являться.

Якщо користувач заповнить усі поля на формі (рис. 3.14), тоді форма пропаде і замість неї з'явиться сторінка, де можна буде створювати запитання (рис. 3.19). Структура цієї сторінки схожа на структуру сторінки, де треба проходити опитування і де можна дивитися результати. У верхній секції сторінки знаходиться заголовок опитування, опис і картинка. Автор опитування мав заповнити ці поля в попередній формі.

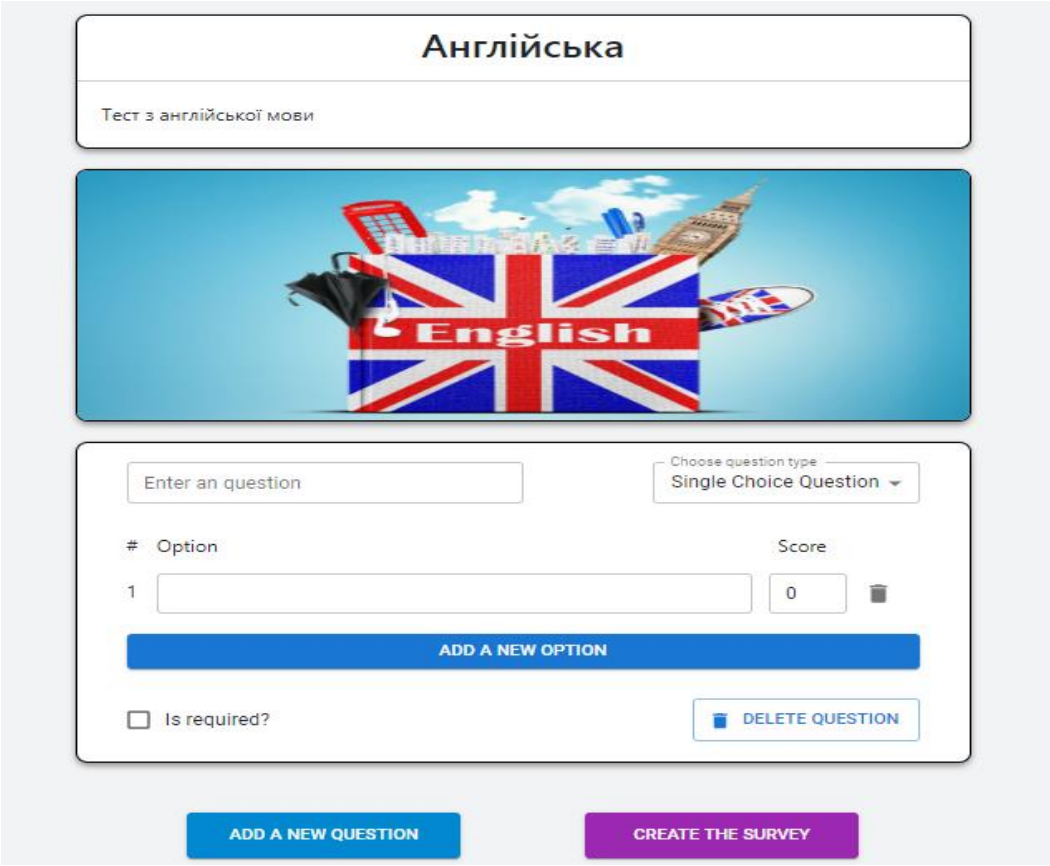


Рисунок 3.19 – Сторінка створення запитань

Основна секція містить запитання, які додав користувач. Спочатку тут знаходиться одне запитання, яке має тип «Single Choice Question», тобто запитання, де потрібно вибрати один варіант відповіді зі списку.

В опитування або тест можна додавати необмежену кількість запитань, для цього потрібно просто натиснути кнопку «Add a new Question» і порожнє поле для запитання з'явиться трохи нижче (рис. 3.20).

Кожне запитання обов'язково містить саму тему запитання, яка вказується в лівому верхньому кутку картки, має один із п'яти типів, який можна вибрати у випадаючому списку у верхньому правому кутку, а також запитання може бути обов'язковим або необов'язковим для відповіді. Щоб зробити запитання обов'язковим, потрібно поставити галочку в нижньому лівому кутку картки. Також можна видалити запитання, натиснувши на кнопку "Delete Question" у правому нижньому кутку картки.

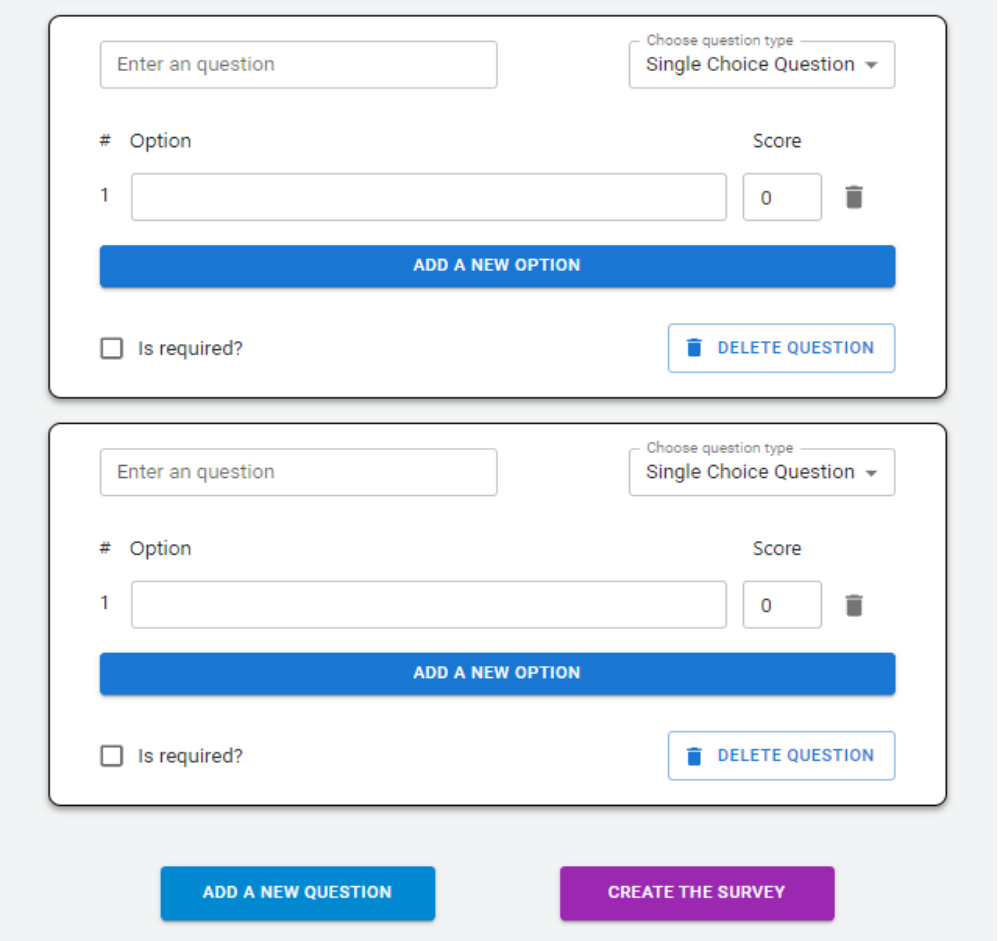


Рисунок 3.20 – Додавання нових запитань у тест

Те, що знаходиться всередині картки, залежить від того, який тип запитання було обрано. Якщо вибрати тип запитання Single Choice Question або Multiple Choice Question, тоді вміст усередині картки буде, як на рис. 3.19. Але при цьому вміст картки також залежить від того, що автор хоче створити – опитування чи тест. Якщо це тест, тоді до кожного варіанта відповіді потрібно вказати ще кількість балів, скільки людина отримає, вибравши певний варіант відповіді. При цьому можна вказувати й від'ємну кількість балів, завдяки чому діє система штрафів за неправильні відповіді, і користувач не зможе в питаннях, де потрібно вибрати кілька варіантів відповіді зі списку, вибрати всі варіанти й отримати максимальний бал. Водночас у поле з балами можна ввести тільки число, якщо спробувати ввести якісь літери, нічого не відбуватиметься.

У запитання типу Single Choice Question і Multiple Choice Question можна додавати необмежену кількість варіантів відповідей, а також можна видаляти ці варіанти відповіді. При цьому нумерація варіантів відповіді змінюватиметься залежно від того, який варіант відповіді було видалено. Тобто якщо ми видалимо варіант під номером 3, тоді наступний варіант, який йшов за цим, стане третім. На рис. 3.21 зображений приклад створення запитання з великою кількістю варіантів відповіді та з від'ємними балами для деяких із варіантів.

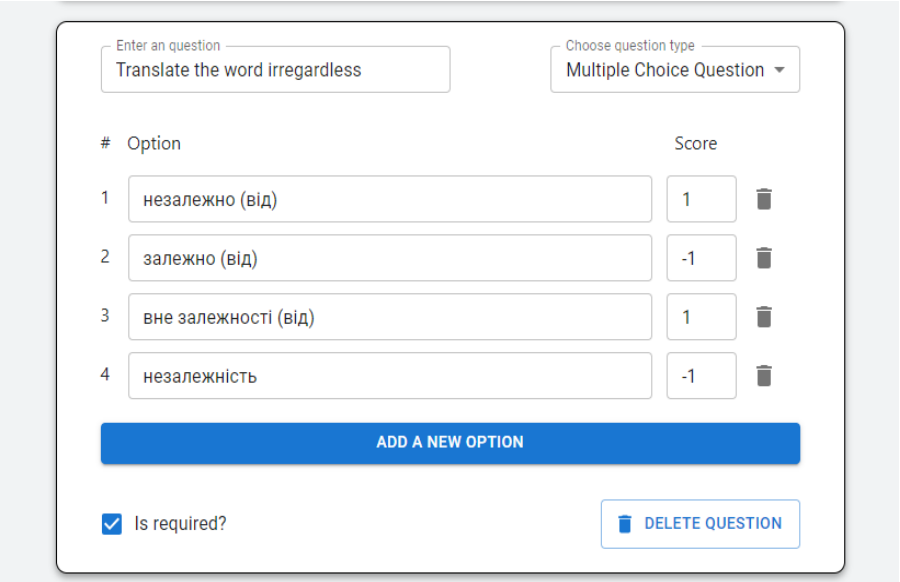


Рисунок 3.21 – Приклад створення запитання з декількома правильними відповідями

Якщо обрати тип запитання Short Answer Question або Detailed Answer Question, тоді треба буде ввести правильну відповідь, а також кількість балів, яку можна за неї заробити, якщо це тест. Крім того, потрібно вказати, чи ігнорувати регістр відповіді респондента, тобто чи має бути точний збіг регістру символів у відповіді респондента з правильною відповіддю.

Якщо це опитування, а не тест, тоді в центральній частині картки взагалі нічого не буде, тобто достатньо буде ввести лише саме запитання і вказати, чи є воно обов'язковим. Різниця між Short Answer Question і Detailed Answer Question лише в розмірі поля, куди респондент вводитиме відповідь, а також у тому, чи буде там прокрутка. Якщо в текстовому полі відповіді в Detailed Answer Question ввести багато тексту, то з'явиться вертикальна прокрутка вниз, що значно спрощує введення великої кількості тексту. Взагалі, бажано використовувати Detailed Answer Question в опитуваннях, у тестах його додавати сенсу немає, але конструктор запитань надає можливість додавати і такий тип запитань у тест. На рис. 3.22 зображено приклад створення запитання з типом Short Answer Question. Щоб переконатися, що можна вказати будь-яку кількість балів за відповідь, вкажемо для цього запитання 3 бали, а потім перевіримо, чи правильно нараховуватимуться бали.

Рисунок 3.22 – Приклад створення запитання з короткою відповідю

Якщо обрати тип запитання Matchmaking Question, тоді всередині картки із запитанням з'являться два списки, до кожного з яких можна додавати нові

варіанти, натиснувши на кнопку «Add a new option». Також нижче є текстове поле, де потрібно ввести кількість балів, яку користувач отримає за одну правильну відповідність. На рис. 3.23 зображено приклад створення такого запитання. Зробимо відповідь на це запитання необов'язковою, щоб потім перевірити, чи правильно працює цей функціонал.

Після натискання на кнопку «Create the survey» буде виконано перехід на головну сторінку сайту, де ми можемо знайти наш створений тест і пройти його. Виконаємо пошуковий запит і знайдемо наш тест на головній сторінці, обравши категорію «Study» (саме цю категорію ми вказали при створенні) і тип Quizes, оскільки при створенні ми вказали, що це буде тест, а не опитування (див. рис. 3.24).

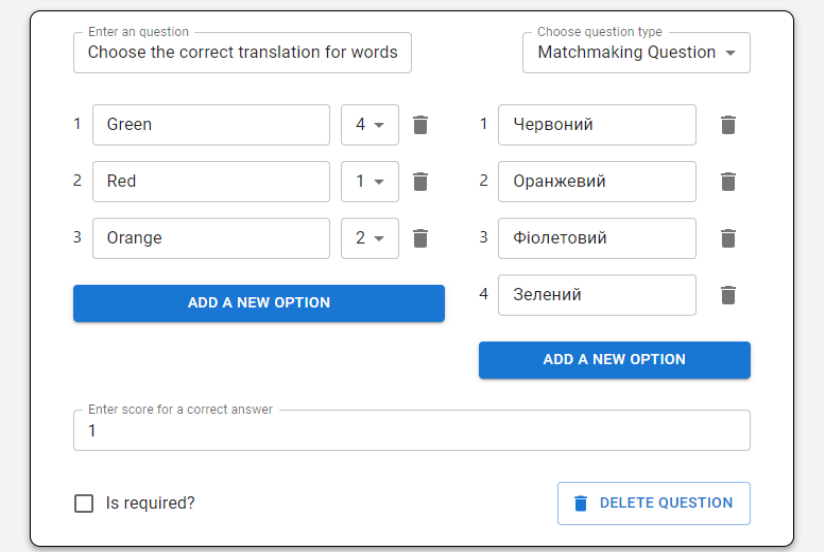


Рисунок 3.23 – Приклад створення запитання на відповідність

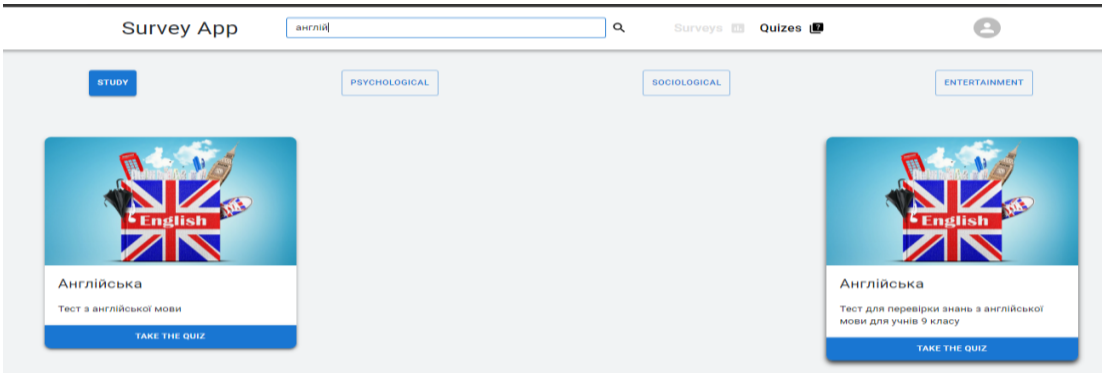


Рисунок 3.24 – Створений тест з’явився на головній сторнці сайту

Натискаємо на кнопку «Take the quiz», даємо якісь відповіді на запитання, але на останнє запитання не відповідаємо, перевіримо, чи правильно працює функціонал обов'язкових запитань.

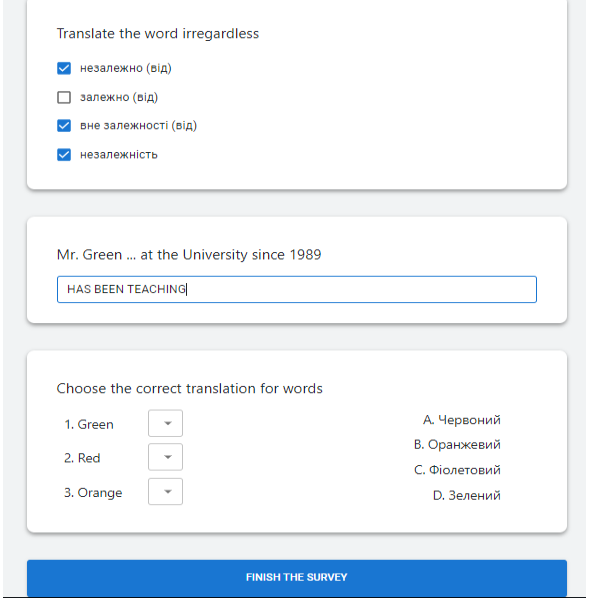


Рисунок 3.25 – Проходження створеного тесту

Після натискання на кнопку "Finish the survey" бачимо, що в результаті ми отримали лише 1 бал із 8 (рис. 3.26).

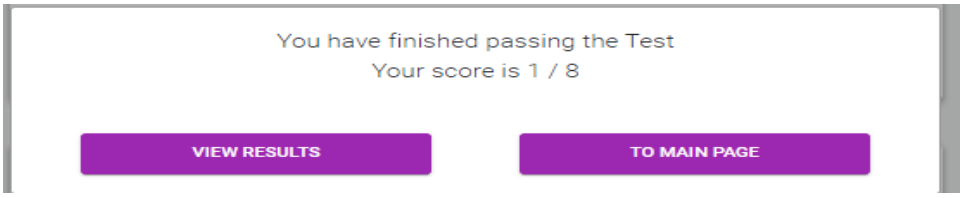


Рисунок 3.26 – Завершення проходження створеного тесту

У першому запитанні ми обрали два правильні варіанти відповіді, за це отримали два бали, але при цьому обрали ще один варіант відповіді, який був неправильним і за нього ми отримали один штрафний бал, у другому запитанні відповідь була правильною, але при створенні цього запитання ми зазначили, що ми не ігноруватимемо регістр відповіді, тобто відповідь повинна була бути написана маленькими літерами, але вона написана великими буквами, тому

відповідь зараховано, як неправильну. На сторінці з результатами проходження тесту немає запитання з відповідниками, оскільки респондент узагалі не дав на нього відповіді, якби він обрав хоча б одну відповідність між варіантами, це запитання з'явилося б у списку. Не дивлячись на те, що цього питання немає в списку, воно буде враховуватися при підрахунку балів, як запитання, за яке користувач отримав 0 балів. (рис. 3.27).

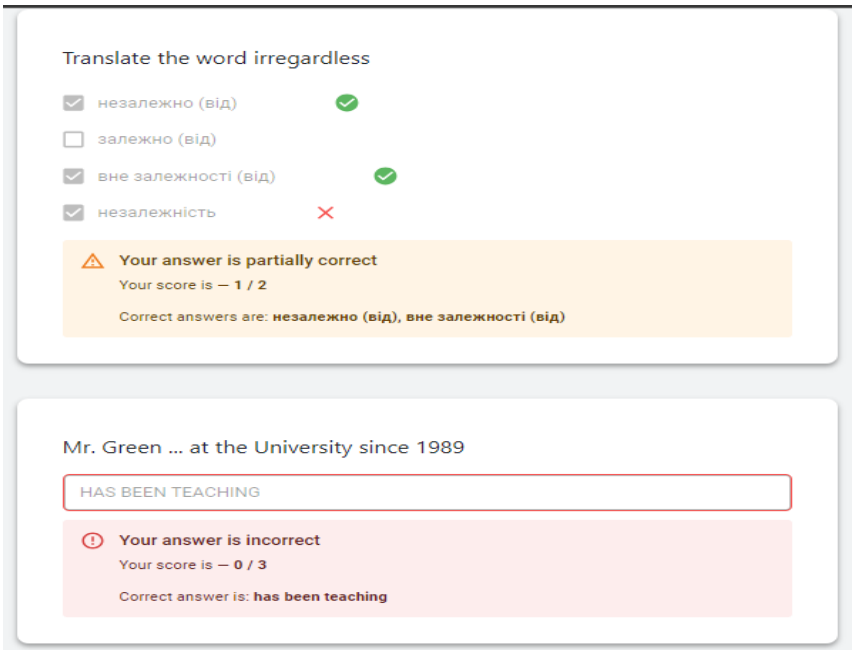


Рисунок 3.27 – Результат проходження створеного тесту

На рис. 3.28 зображений код функції, яка підраховує кількість балів, які респондент заробив за проходження тесту. Функція приймає як параметр список відповідей, які дав користувач answersToQuestions, потім проходить по всіх відповідях і підсумовує кількість балів, які користувач отримав за кожну відповідь. Якщо його відповідь – це набір варіантів (запитання з кількома правильними відповідями), тоді береться кожен варіант і визначається кількість балів, які респондент отримав за цей варіант, потім ці бали підсумовуються. При цьому, варіант може мати від'ємну кількість очок, через що може знизити загальну оцінку. Функція calculateEarnedScore викликається після завершення тесту.

```
export const calculateEarnedScore = (answersToQuestions: IAnswerToQuestion[]): number => {
    let earnedScore: number = 0;

    answersToQuestions.forEach(answerToQuestion => {
        const answer = answerToQuestion.answer;

        if (isOption(answer) || isTextAnswer(answer)) {
            if (answer.score) {
                earnedScore += answer.score;
            }
        } else if (isMatches(answer)) {
            answer.leftList.forEach(option => {
                if (option.score) {
                    earnedScore += option.score;
                }
            })
        } else if (isSetOfOptions(answer)) {
            answer.forEach(option => {
                if (option.score) {
                    earnedScore += option.score;
                }
            })
        }
    })

    return earnedScore;
}
```

Рисунок 3.28 – Код функції, яка підраховує кількість балів

Крім зароблених балів, також треба визначити максимальний бал за тест, який респондент може отримати. Максимальний бал визначається в момент створення тесту, немає сенсу визначати максимальну кількість балів після проходження тесту, тому що ця процедура буде повторюватися безліч разів, але результат буде той самий. За підрахунок максимальної кількості балів за тест відповідає функція calculateMaximumScore (рис. 3.29), яка приймає параметром список запитань questions, проходить кожне запитання і визначає максимальну кількість балів, які можна за нього заробити. При цьому підсумовуються тільки позитивні бали, тобто якщо у запитання є список варіантів, деякі з яких мають негативний бал, такі варіанти не враховуються під час підрахунку.

Під час розробки ми зіткнулися з проблемою, коли сторінка, де можна було створювати питання для тестів і опитувань, працювала дуже повільно. Це відбувалося через те, що при кожній зміні текстових полів, перемальовувалася вся сторінка. Взагалі, додатки, написані на React, зазвичай вирізняються швидкістю роботи завдяки використанню концепції віртуального DOM, але в деяких випадках, питаннями оптимізації швидкості сторінки доводиться займатися розробнику самостійно. І в процесі розробки ми зіткнулися з таким випадком. У React є хук useMemo, завдяки цьому хуку React може запам'ятати

стан якихось елементів сторінки, а потім замість того, щоб перемальовувати їх щоразу, займатися їхнім рендерингом тільки тоді, коли вони дійсно змінюються.

```
export const calculateMaximumScore = (questions: IQuestion[]) => {
    let maximumScore: number = 0;

    questions.forEach(question => {
        if (
            question.type === QuestionType.OneChoice ||
            question.type === QuestionType.MultipleChoice
        ) {
            if (isSetOfOptions(question.options)) {
                question.options.forEach(option => {
                    if (option.score !== undefined && option.score > 0) {
                        maximumScore += option.score as number;
                    }
                })
            }
        } else if (
            question.type === QuestionType.ShortTextField ||
            question.type === QuestionType.DetailedTextField
        ) {
            if (
                isTextAnswer(question.correctAnswer)
                && question.correctAnswer.score !== undefined
                && question.correctAnswer.score > 0
            ) {
                maximumScore += question.correctAnswer.score as number;
            }
        } else if (question.type === QuestionType.Matchmaking) {
            if (isMatches(question.options)) {
                question.options.leftList.forEach(option => {
                    if (option.score !== undefined && option.score > 0) {
                        maximumScore += option.score as number;
                    }
                })
            }
        }
    })

    return maximumScore;
}
```

Рисунок 3.29 – Код функції, яка підраховує максимальний бал за тест

Наприклад, якщо у нас є багато текстових полів на сторінці сайту (у нашому випадку це сторінка, де можна створювати запитання) і ми змінюємо текст в одному з полів, React буде перемальовувати всю сторінку повністю та всі текстові поля разом із текстом, який там міститься, а це досить трудомістка операція. Щоб уникнути цього, можемо використовувати useMemo і серед усіх текстових полів на сайті будемо перемальовувати тільки ті, де дійсно було змінено текст.

На рис. 3.30 зображений фрагмент коду компонента TextQuestionConstructor, який відповідає за створення запитань, де користувачеві потрібно дати текстову відповідь.

```
return (
    <div className={style.TextQuestionConstruct} style={cssProperties}>
        {
            useMemo(
                () => renderCorrectAnswerTextField(question.correctAnswer),
                [question?.correctAnswer?.text]
            )
        }

        {
            useMemo(
                () => renderScoreField(question.correctAnswer),
                [question?.correctAnswer?.score]
            )
        }

        {
            useMemo(
                () => renderFooter(question.correctAnswer),
                [question?.correctAnswer?.ignoreRegister]
            )
        }
    </div>
);
```

Рисунок 3.30 – Фрагмент коду компонента TextQuestionConstructor

У конструкторі текстових запитань ми вказуємо правильну відповідь, кількість балів за правильну відповідь, а також те, чи ігнорується регістр відповіді. Усього три поля для заповнення, за відмальовування кожного з цих полів відповідає своя функція, будемо запам'ятовувати, що записано в цих полях за допомогою useMemo, і перемальовувати ці поля тільки в тому разі, якщо ми дійсно змінили значення в якомусь із цих полів. Аналогічно useMemo був використаний і для компонентів, які відповідали за створення запитань інших типів. Завдяки цьому, вийшло сильно підвищити швидкість роботи сторінки, де створювалися опитування і тести.

3.2 Тестування

Для тестування додатка будемо використовувати бібліотеку Jest. Ця технологія є найпопулярнішим рішенням для тестування JavaScript додатків. За допомогою цієї бібліотеки напишемо Unit тести для основного функціоналу додатка: підрахунок максимальної оцінки за тест, підрахунок оцінки, яку отримав респондент за тест, а також створення коментаря на відповідь респондента. Також опишемо загальну схему, як можна буде додавати нові тести до нового функціоналу.

Архітектура проєкту влаштована таким чином, що вся бізнес-логіка ізольована від тієї частини застосунку, яка відповідає за користувацький інтерфейс. Здебільшого всі React компоненти містять тільки функції, які виконують рендеринг якихось елементів, наприклад, рендеринг кнопок, текстових полів тощо, при цьому функції, які виконують складну логіку, підключаються ззовні. Завдяки цьому, можна легко проводити тестування цих функцій, тому що вони не мають залежностей із компонентами, які відповідають за користувацький інтерфейс.

У package.json файлі проєкту (рис. 3.31) є скрипт, який автоматично шукає і виконує всі тести, які є в проєкті. Цей скрипт можна запустити, використовуючи команду npm run test.

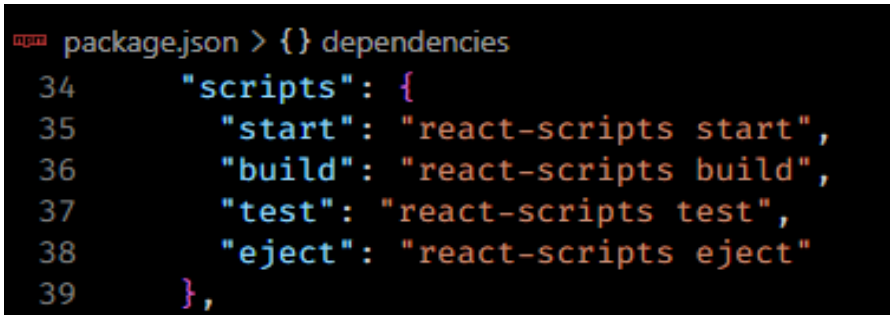


Рисунок 3.31 – Скрипт для тесту у файлі package.json

У проєкті є директорія tests, куди поміщаються всі тести. Кожен тест – це файл із назвою функції, для якої створюється тест, і розширенням .test.ts.

На рис. 3.32 зображений вміст цієї теки. Оскільки тестувати будемо тільки три функції, файлів буде теж три.

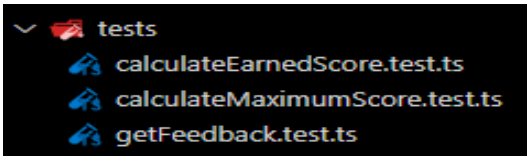


Рисунок 3.32 – Вміст теки для тестів

Для додавання нового тесту, потрібно просто додати новий файл з відповідною назвою в цю директорію. Взагалі, назва файлу з тестом не має

обов'язково збігатися з назвою функції, але для зручності бажано зробити саме так, щоб у майбутньому можна було легко підправити якісь тести і не довелося довго шукати потрібний. Приклади того, який вміст повинен бути у файлів з тестами, будуть описані далі.

Для тестування функції підрахунку балів, які користувач заробив за тест, використовуємо набір даних, який зображено на рис. 3.33. Ми розглядаємо три можливі випадки. У першому випадку, всі відповіді, які дав респондент, будуть неправильними. У другому випадку, відповіді респондента будуть частково правильними, а в третьому – усі відповіді респондента правильні.

```
const option1: IOption = {id: 1, label: 'Бігти', score: 1};
const option2: IOption = {id: 2, label: 'Ходити', score: -1};
const option3: IOption = {id: 3, label: 'Пригати', score: -1};
const option4: IOption = {id: 4, label: 'Керувати', score: 1};
const correctAnswer: ITextAnswer = {text: 'exhausted', score: 1, ignoreRegister: false};

const question1: IQuestion = {
    id: 1, topic: 'Перекладіть слово run', type: QuestionType.MultipleChoice,
    options:  [option1, option2, option3, option4], required: true
};
const question2: IQuestion = {
    id: 2, topic: `When I finish work, I'm usually ... (exhaust)`,
    type: QuestionType.ShortTextField, correctAnswer: correctAnswer, required: true
};

const incorrectAnswersToQuestions: IAnswerToQuestion[] = [
    { question: question1, answer: [option2, option3] },
    { question: question2, answer: {text: 'exhausting', score: 0} }
];

const partiallyCorrectAnswersToQuestions: IAnswerToQuestion[] = [
    { question: question1, answer: [option1, option2] },
    { question: question2, answer: {text: 'exhausted', score: 1} }
];

const correctAnswersToQuestions: IAnswerToQuestion[] = [
    { question: question1, answer: [option1, option4] },
    { question: question2, answer: {text: 'exhausted', score: 1} }
];
```

Рисунок 3.33 – Набір даних для функції підрахунку балів, які заробив респондент

Ми заздалегідь знаємо, який результат ми хочемо отримати в результаті роботи цієї функції. Тому нам треба передати кожен із цих наборів у функцію, порівняти бажаний результат, який ми хочемо отримати, з реальним результатом, який поверне функція. Якщо фактичний результат збігається з тим, який ми очікуємо, це означає, що тест пройдено. На рис. 3.34 ми описуємо, які результати ми чекаємо від функції.

```
describe('Calculate earned score', () => {
    test('Incorrect answer', () => {
        expect(calculateEarnedScore(incorrectAnswersToQuestions)).toBe(-2);
    })
    test('Partially correct answer', () => {
        expect(calculateEarnedScore(partiallyCorrectAnswersToQuestions)).toBe(1);
    })
    test('Correct answer', () => {
        expect(calculateEarnedScore(correctAnswersToQuestions)).toBe(3);
    })
})
```

Рисунок 3.34 – Очікуваний результат функції підрахунку балів, які заробив респондент

Аналогічно, опишемо набори даних і бажані результати для інших двох функцій, які ми будемо тестувати. На рис. 3.35 зображено набір даних для тестування функції, яка обчислюватиме для тесту максимальний бал, який може заробити за цей тест респондент. Опишемо три можливих випадки: у першому випадку, передамо у функцію запитання, у яких немає правильної відповіді взагалі, у другому випадку це будуть запитання, які мають штрафи за неправильні відповіді, а в третьому випадку це будуть запитання, де немає штрафів за неправильні відповіді.

```
const questionsWithoutCorrectAnswers: IQuestion[] = [
    {
        id: 1, topic: 'Квадратний корінь із 196', type: QuestionType.ShortTextField, required: true,
        correctAnswer: {text: '14', ignoreRegister: true, score: 0} as ITextAnswer
    },
    {
        id: 2, topic: 'Столиця Іспанії', type: QuestionType.OneChoice, required: true,
        options: [{id: 1, label: 'Мадрид', score: 0}, {id: 2, label: 'Пекін', score: -1}]
    }
];

const questionsWithPenaltiesForWrongAnswers: IQuestion[] = [
    {
        id: 1, topic: 'Квадратний корінь із 196', type: QuestionType.ShortTextField, required: true,
        correctAnswer: {text: '14', ignoreRegister: true, score: 1} as ITextAnswer
    },
    {
        id: 2, topic: 'Столиця Іспанії', type: QuestionType.OneChoice, required: true,
        options: [{id: 1, label: 'Мадрид', score: 1}, {id: 2, label: 'Пекін', score: -1}]
    }
];

const questionsWithoutPenaltiesForWrongAnswers: IQuestion[] = [
    {
        id: 1, topic: 'Квадратний корінь із 196', type: QuestionType.ShortTextField, required: true,
        correctAnswer: {text: '14', ignoreRegister: true, score: 1} as ITextAnswer
    },
    {
        id: 2, topic: 'Столиця Іспанії', type: QuestionType.OneChoice, required: true,
        options: [{id: 1, label: 'Мадрид', score: 1}, {id: 2, label: 'Пекін', score: 0}]
    }
]
```

Рисунок 3.35 – Набір даних для функції, яка рахує максимальний бал за тест

На рис. 3.36 можна побачити, які результати має повернути функція в кожному з цих випадків.

```
describe('Calculating the maximum score for a test', () => {
    test('Questions without correct answers', () => {
        expect(calculateMaximumScore(questionsWithoutCorrectAnswers)).toBe(0);
    })
    test('Questions with penalties for wrong answers', () => {
        expect(calculateMaximumScore(questionsWithPenaltiesForWrongAnswers)).toBe(2);
    })
    test('Questions without penalties for wrong answers', () => {
        expect(calculateMaximumScore(questionsWithoutPenaltiesForWrongAnswers)).toBe(2);
    })
})
```

Рисунок 3.36 – Очікуваний результат функції, яка рахує максимальний бал за тест

Тут важливо зауважити, що якщо правильних відповідей у тесті немає, тоді функція поверне 0. У тій частині кода додатку, яка відповідає за користувацький інтерфейс, є перевірка на це. Якщо автор спробує створити тест, де немає правильних відповідей, він отримає повідомлення про помилку (рис. 3.37). Проведемо ручне тестування, щоб переконатися, що ця перевірка справді працює і що в автора не вийде створити тест, де немає правильних відповідей.

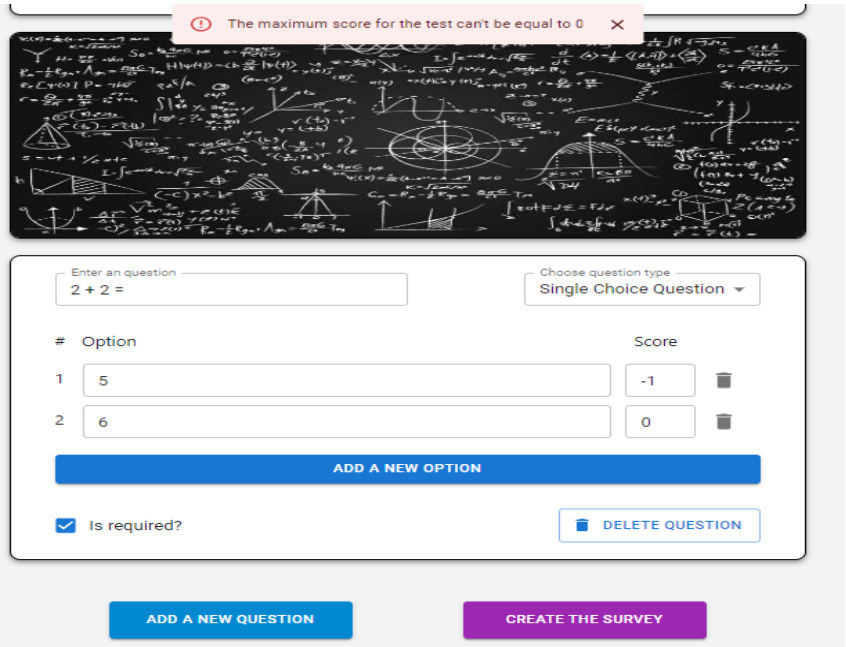


Рисунок 3.37 – Повідомлення про помилку, що в тесті немає правильних відповідей

Далі протестуємо функцію, яка створює коментар для відповіді на запитання. Візьмемо запитання, де можна вибрати кілька варіантів відповіді, при цьому за неправильну відповідь нараховується штрафний бал. Опишемо три випадки: коли відповідь правильна, коли відповідь частково правильна і коли відповідь неправильна (рис. 3.38).

```
const option1 = {id: 1, label: '3', score: 1};
const option2 = {id: 2, label: '-3', score: 1};
const option3 = {id: 3, label: '3.5', score: 0};

const question: IQuestion =  {
    id: 1,
    topic: 'x^2 = 9',
    type: QuestionType.MultipleChoice,
    required: true,
    options: [option1, option2, option3]
} as IQuestion;

const incorrectAnswerToQuestion: IAnswerToQuestion = {
    question: question,
    answer: [option3] as IAnswer
};

const partiallyCorrectAnswerToQuestion: IAnswerToQuestion = {
    question: question,
    answer: [option1, option3] as IAnswer
};

const correctAnswerToQuestion: IAnswerToQuestion = {
    question: question,
    answer: [option1, option2] as IAnswer
};
```

Рисунок 3.38 – Набір даних для функції створення коментаря

Тепер опишемо, які результати ми чекаємо після виконання функції, якщо ми передамо туди ці вхідні дані (рис. 3.39).

```
describe('Get feedback', () => {
    test('Incorrect answer', () => {
        expect(getFeedback(incorrectAnswerToQuestion)).toEqual({
            totalScore: -1, maximumScore: 2, correctAnswers: ['3', '-3']
        });
    })
    test('Partially correct answer', () => {
        expect(getFeedback(partiallyCorrectAnswerToQuestion)).toEqual({
            totalScore: 0, maximumScore: 2, correctAnswers: ['3', '-3']
        })
    })
    test('Correct answer', () => {
        expect(getFeedback(correctAnswerToQuestion)).toEqual({
            totalScore: 2, maximumScore: 2, correctAnswers: ['3', '-3']
        })
    })
})
```

Рисунок 3.39 – Очкуваний результат для функції створення коментаря

Результат виконання всіх тестів зображений на рис. 3.40.

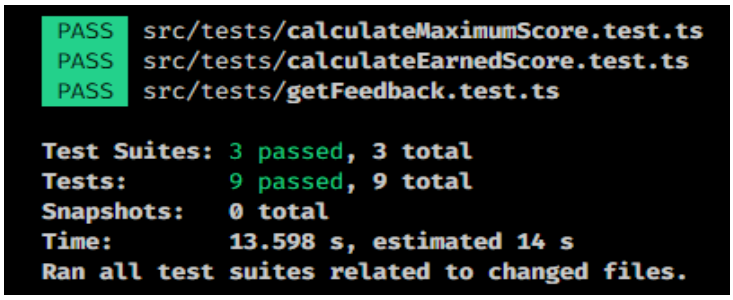


Рисунок 3.40 – Результат виконнаня всіх тестів

ВИСНОВКИ

У роботі був розроблений веб-сервіс для проведення опитувань і тестів. Розроблений сервіс має такий функціонал: створення тестів і опитувань, можливість проходити їх і дивитися результати проходження разом з автоматичною оцінкою (якщо це тест), пошук і фільтрація тестів і опитувань на головній сторінці, реєстрація та авторизація, можливість редагувати, видаляти тести та опитування, переглядати результати проходження тестів та опитувань для автора.

У першому розділі були описані функціональні та нефункціональні вимоги до розроблюваного застосунку, був наведений огляд уже існуючих сервісів для проведення тестів і опитувань, виконано порівняльний аналіз цих застосунків із тим, який розробляється в роботі.

У другому розділі були побудовані функціональні схеми застосунку для різних користувачів сервісу залежно від їхньої ролі (респондент і автор), була побудована діаграма компонентів, а також ER діаграма та діаграма послідовності. Також у цьому розділі був описаний архітектурний патерн Redux з обґрунтуванням, чому саме цей патерн був обраний для розробки.

У третьому розділі був описаний і продемонстрований функціонал додатка, також були описані важливі моменти з розробки, були представлені приклади коду. Також цей розділ містить автоматизоване та ручне тестування деяких основних функцій застосунку разом із поясненням, як можна додавати нові тести.

Підсумовуючи, React, Redux і Firebase – хороші інструменти для створення веб-додатків. Результати роботи можуть бути використані для подальшого використання у сфері розробки веб-додатків, а також у сфері розробки додатків для проведення тестів і опитувань.

ПЕРЕЛІК ПОСИЛАНЬ

1. Сергієнко В. П., Кухар Л. О. Методичні рекомендації зі складання тестових завдань. Міністерство освіти і науки, молоді та спорту України. Національний університет імени М. П. Драгоманова. URL: https://moodle-student.udu.edu.ua/pluginfile.php/32/mod_resource/content/1/Методичні%20рекомендації%20зі%20складання%20тестових%20завдань.pdf (дата зверенення: 18.04.2023).
2. Online Test Pad – Онлайн тести, опитування, кросворди. Онлайн конструктор тестів, опитувань, кросвордів. URL: https://onlinetestpad.com/ (дата звернення: 12.04.2023).
3. Simpoll – Сервіс опитувань. URL: https://simpoll.ru/ (дата звернення: 14.03.2023).
4. Survio. Free surveys and questionnaires online. URL: https://www.survio.com/ (дата звернення: 16.03.2023).
5. Метод аналізу ієрархій Т. Сааті – Моделювання систем і процесів – Підручники для студентів онлайн. URL: https://stud.com.ua/25063/menedzhment/metod_analizu_iyerarhiy_saati (дата звернення: 17.04.2023).
6. Passing Props to a Component – React. URL: https://react.dev/learn/passing-props-to-a-component (дата звернення: 17.04.2023).
7. Structure Your Database. Firebase Realtime Database. URL: https://firebase.google.com/docs/database/web/structure-data (дата звернення: 18.04.2023).
8. React.js Architecture Pattern: Implementation + Best Practices. URL: https://www.knowledgehut.com/blog/web-development/react-js-architecture (дата звернення: 19.04.2023).
9. Redux Fundamentals, Part 4: Store. Redux. URL: https://redux.js.org/tutorials/fundamentals/part-4-store#dispatching-actions

(дата звернення: 19.04.2023).

10. MUI: The React Component Library you always wanted. URL: https://mui.com (дата звернення: 20.04.2023).