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 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.Ю. Саранча
(ініціали та прізвище)

Керівник професор кафедри комп'ютерних наук, професор, д.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

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)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Саранчі Владиславу Юрійовичу
(прізвище, ім’я та по-батькові)

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

керівник роботи Чопоров Сергій Вікторович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від «21» грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 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. Проєктування та розробка вебсервісу для тестування.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 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.	09.01.2024	
2.	Збір вихідних даних.	30.01.2024	
3.	Обробка методичних та теоретичних	20.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	19.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2024	

Студент _____ В.Ю. Саранча
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.В. Чопоров
(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»: 50 с., 17 рис., 10 джерел.

БАЗА ДАНИХ, ВЕБСЕРВІС, КОМПОНЕНТ, КОНТРОЛЕР, МОДЕЛЬ, ФРЕЙМВОРК, FIREBASE, REACT, REDUX.

Об’єкт дослідження – процес розробки вебсервісу для проведення опитувань та тестувань.

Мета роботи: розробка вебсервісу для проведення опитувань та тестувань.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії.

Предмет дослідження – розробка вебсервісу для проведення опитувань та тестувань з використанням фреймворків React, Redux, та Firebase.

У роботі наведено етапи розробки вебсервісу для проведення опитувань та тестувань з використанням фреймворків React, Redux, та Firebase, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Описано предметну область. Розроблено вебсервіс для проведення опитувань та тестувань з використанням фреймворків React, Redux, та Firebase, який використовується для організації опитувань і тестувань, а також для збору і аналізу даних у реальному часі. Зростання попиту на інтерактивні онлайн сервіси робить важливим розв’язок таких задач, розгортання та підтримки вебсервісів для проведення опитувань та тестувань. Тому, розробка вебсервісу для проведення опитувань та тестувань з використанням фреймворків React, Redux, та Firebase є актуальною задачею.

SUMMARY

Bachelor's qualifying paper "Development of a Web Service for Conducting Surveys and Tests Using React, Redux, Firebase": 50 pages, 17 figures, 10 references.

COMPONENT, CONTROLLER, DATABASE, MODEL, WEB SERVICE, FIREBASE, FRAMEWORK, REACT, REDUX.

Research Object – is the process of developing a web service for conducting surveys and testing.

Objective of the Thesis: development of a web service for conducting surveys and testing.

Research Methods – are object-oriented programming methods, software engineering methods.

Research Subject – is development of a web service for conducting surveys and testing using React, Redux, and Firebase frameworks.

The thesis outlines the stages of developing a web service for conducting surveys and testing using React, Redux, and Firebase, identifies key issues, and applies new methods for gathering and analyzing software requirements. The domain area is described. A web service for conducting surveys and testing using React, Redux, and Firebase has been developed, utilized for organizing surveys and testing, as well as for real-time data collection and analysis. The increasing demand for interactive online services makes the deployment and support of web services for conducting surveys and testing critical. Therefore, the development of a web service for conducting surveys and testing using React, Redux, and Firebase remains a relevant task.

ЗМІСТ

ВСТУП

Розвиток сучасних технологій інтенсивно перетворює різні аспекти нашого життя, змінюючи способи взаємодії та надання послуг через Інтернет. В сучасному цифровому світі особливо важливо, щоб вебсервіси пропонували не лише ефективні, але й інтуїтивно зрозумілі інструменти для проведення опитувань та тестувань. Споживачі очікують миттєвої відповіді та швидкої реакції на їхні запити, що стає ключовим чинником для підтримки високого рівня обслуговування та поліпшення користувацького досвіду.

Вебсервіси сьогодні є критично важливою частиною будь-якого бізнесу, забезпечуючи основний механізм спілкування з клієнтами та користувачами. У конкурентному середовищі важливо мати інструменти, які дозволяють реальну взаємодію в режимі реального часу, надаючи необхідну підтримку та оперативно відповідаючи на запити.

Даний проєкт спрямований на розробку вебсервісу для проведення опитувань та тестувань з використанням передових технологій: React для створення інтерфейсу, Redux для управління станом додатку та Firebase для зберігання даних та реального часу. Метою проєкту є створення ефективного інструменту, який забезпечить швидку та зручну взаємодію з користувачами, адаптуючись до їхніх потреб і забезпечуючи безпеку та надійність обміну даними.

В рамках цього проєкту будуть розроблені компоненти інтерфейсу та логіка додатку з використанням React та Redux для оптимальної продуктивності і управління станом. Firebase надасть можливості для швидкого зберігання та маніпулювання даними, що сприятиме зручному керуванню опитуваннями та тестуваннями. Застосування сучасних засобів безпеки допоможе забезпечити захист конфіденційної інформації користувачів.

Цей проєкт не лише покликаний покращити якість сервісу для клієнтів, але й сприяти підвищенню ефективності взаємодії з користувачами в онлайн-середовищі. Використання React, Redux та Firebase дозволяє створити інтерактивне середовище, що відповідає вимогам сучасного ринку та сприяє підвищенню задоволення від використання вебсервісу.

Таким чином, розробка вебсервісу для проведення опитувань та тестувань з використанням React, Redux та Firebase є кроком у напрямку створення інноваційних і надійних вебзастосунків, які відповідають потребам сучасного користувача та вимогам ринку.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБСЕРВІСУ

1.1 Огляд сучасних технологій у розробці вебсервісів

У сучасному світі інформаційні технології займають центральне місце в різних сферах життя. Одним з важливих напрямків розвитку IT є створення вебсервісів для проведення опитувань та тестувань. Це зумовлено необхідністю швидкого та ефективного збору даних, а також оцінки знань і компетенцій у різних контекстах – від освіти до бізнесу та громадських досліджень [3].

Однією з головних причин актуальності вебсервісів для опитувань є потреба у швидкому та ефективному зборі даних. У бізнесі, наприклад, компанії використовують такі сервіси для отримання зворотного зв’язку від клієнтів, оцінки задоволеності продуктами або послугами, а також для вивчення ринку. Це дозволяє оперативно реагувати на зміни в потребах клієнтів і відповідно коригувати стратегії.

Вебсервіси для опитувань та тестувань мають низку технологічних переваг. Вони забезпечують доступність з будь-якої точки світу, де є інтернет, що значно розширює коло потенційних учасників. Крім того, ці сервіси часто інтегруються з іншими програмними продуктами, такими як CRM-системи, системи управління навчанням (LMS) та інші, що робить їх ще більш функціональними та зручними у використанні [8].

Google Forms є одним з найбільш відомих і широко використовуваних вебсервісів для створення опитувань та тестів [1].

Ключові функції та переваги:

– легкість використання: інтуїтивно зрозумілий інтерфейс дозволяє швидко створювати форми без необхідності спеціальних навичок;
– інтеграція з Google Workspace: форми легко інтегруються з іншими інструментами Google, такими як Google Sheets для аналізу результатів;

– безкоштовність: сервіс є безкоштовним, що робить його доступним для широкої аудиторії;
– гнучкість налаштувань: можливість додавання різних типів питань (текстових, варіантів відповіді, шкал оцінок тощо).

Google Forms – це потужний інструмент для створення та проведення опитувань і тестувань, який не потребує спеціальних знань у програмуванні. Він забезпечує зручний спосіб збору та аналізу даних, що робить його ідеальним вибором для освітніх, дослідницьких та бізнес-завдань (див. рис. 1.1).

Рисунок 1.1 – Google Forms [2]

SurveyMonkey – це потужний інструмент для створення та аналізу опитувань, популярний серед бізнес-користувачів та дослідників.

Ключові функції та переваги:

– широкий набір шаблонів: надає велику кількість готових шаблонів для різних типів опитувань;
– аналітичні можливості: потужні інструменти для аналізу даних, включаючи побудову звітів та візуалізацію результатів;
– інтеграція з іншими сервісами: можливість інтеграції з CRM-системами, маркетинговими платформами та іншими інструментами;

– платні та безкоштовні плани: доступність безкоштовного плану з базовими функціями, а також розширених платних планів.

SurveyMonkey – це потужний та гнучкий інструмент для створення та проведення опитувань. Він забезпечує багатий функціонал для збору даних та їх аналізу, що робить його ідеальним вибором для професіоналів у різних галузях, включаючи маркетинг, освіту, HR та багато інших (див. рис. 1.2).

Рисунок 1.2 – SurveyMonkey [3]

Typeform виділяється своїм унікальним підходом до створення форм та опитувань, пропонуючи інтерактивний і візуально привабливий досвід для користувачів [10].

Ключові функції та переваги:

– інтерактивний дизайн: форми та опитування створені для того, щоб бути цікавими та залучати користувачів до взаємодії;

– гнучкі налаштування: можливість створення різних типів питань з умовною логікою, що дозволяє налаштовувати унікальні маршрути для респондентів;

– інтеграція: легка інтеграція з багатьма інструментами, такими як Google Sheets, Mailchimp, Slack тощо;

– аналітика: зручні інструменти для аналізу та експорту даних.

Typeform – це потужний інструмент для створення інтерактивних та адаптивних форм опитувань і тестувань. Він забезпечує відмінний користувацький досвід, що робить його ідеальним для маркетингових досліджень, збору зворотного зв'язку, навчальних опитувань та багатьох інших завдань. Завдяки своїй гнучкості та інтеграціям з різними сервісами Typeform дозволяє ефективно збирати та аналізувати дані (див. рис. 1.3).

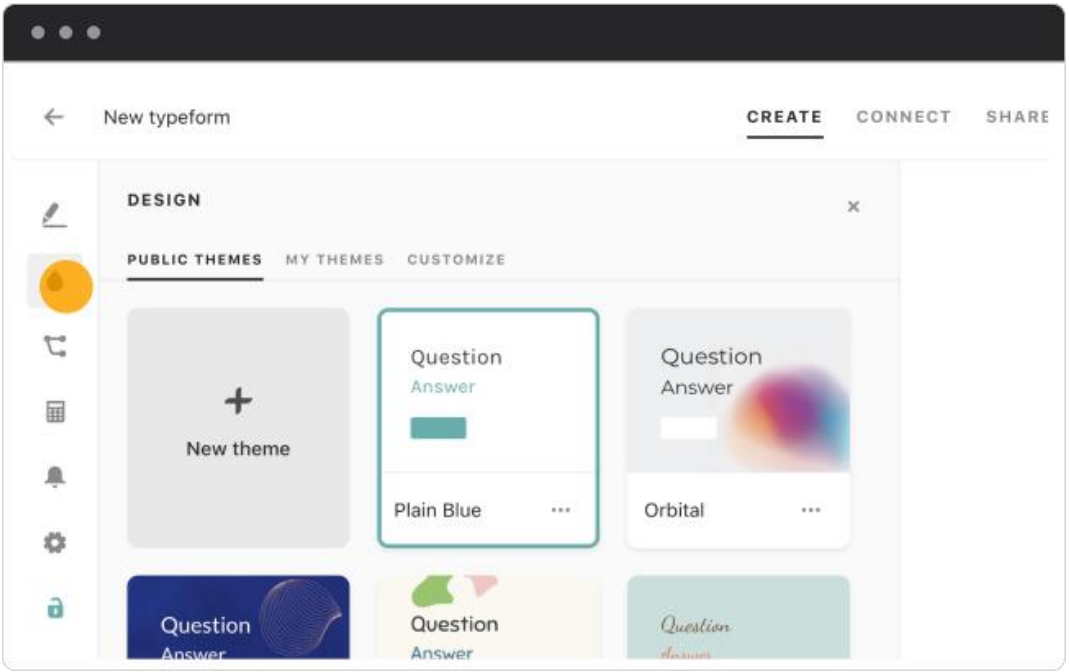

Рисунок 1.3 – Typeform [5]

Kahoot – це платформа для створення ігрових тестів та вікторин, що використовується в основному в освітніх закладах.

Ключові функції та переваги:

– ігровий формат: тести представлені у вигляді гри, що робить навчання цікавим і захоплюючим;

– мультиплеєр: можливість проведення тестів в режимі реального часу з кількома учасниками;

– візуальні та звукові ефекти: використання мультимедіа для залучення уваги учасників;

– мобільний додаток: доступність мобільного додатку для легкого доступу до тестів з будь-якого місця.

Kahoot – це ефективний інструмент для створення інтерактивних навчальних ігор, що робить навчання цікавим та залучаючим. Завдяки своїй простоті використання та потужним функціям, Kahoot ідеально підходить для освітніх і корпоративних цілей, допомагаючи підтримувати високу мотивацію учасників та забезпечуючи ефективне засвоєння матеріалу (див. рис. 1.4).

Рисунок 1.4 – Kahoot [5]

Qualtrics – це професійний інструмент для створення опитувань, який широко використовується в бізнесі та наукових дослідженнях [7].

Ключові функції та переваги:

- розширені аналітичні можливості: потужні інструменти для збору та аналізу даних, включаючи машинне навчання та прогнозну аналітику;
- інтеграція з бізнес-процесами: легка інтеграція з різними бізнес-системами та програмами;
- гнучкість у налаштуванні: можливість створення складних анкет з умовною логікою та багатьма варіантами відповідей;
- підтримка різних платформ: доступність для використання на різних пристроях, включаючи мобільні.

Qualtrics – це універсальний інструмент для проведення опитувань та досліджень, що дозволяє легко створювати, розповсюджувати та аналізувати опитування. Завдяки своїм потужним функціям та інтеграціям, Qualtrics ідеально підходить для бізнесу, наукових досліджень та освітніх установ, допомагаючи отримувати цінні інсайти та покращувати прийняття рішень на основі зібраних даних (див. рис. 1.5).

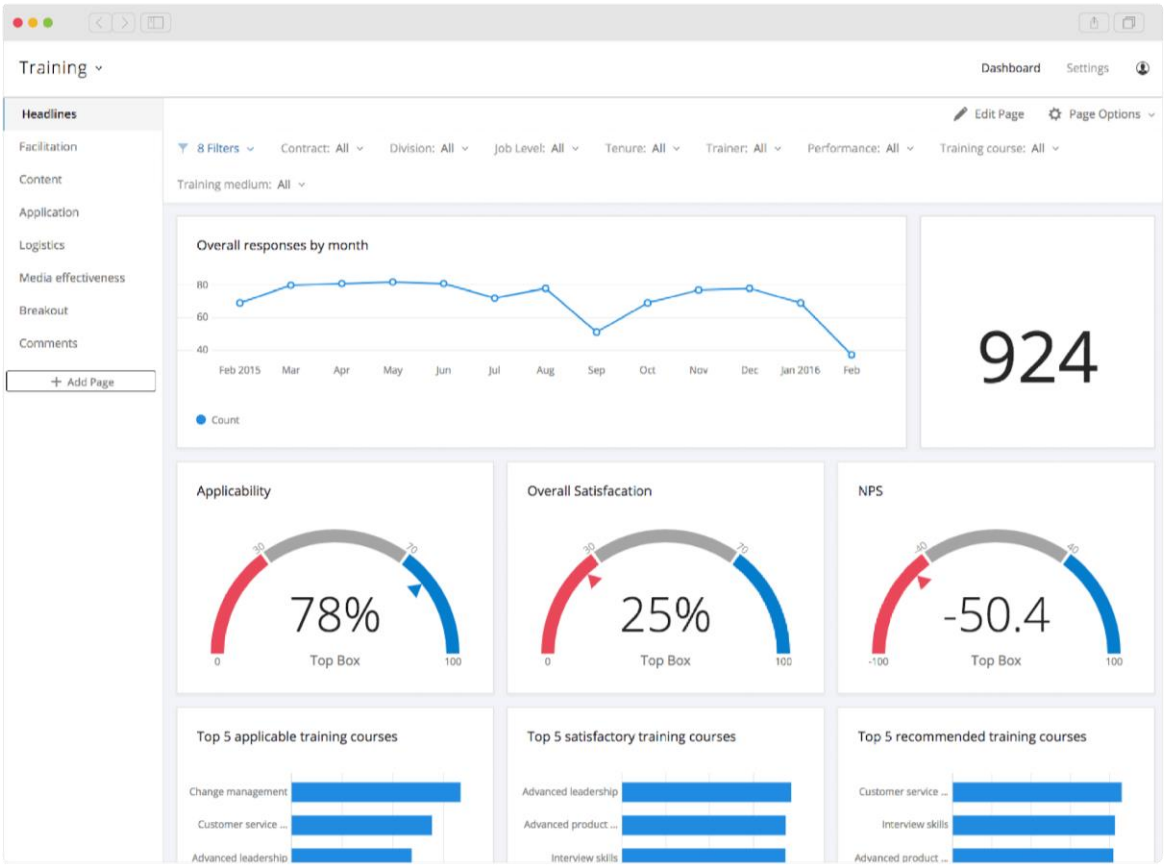

Рисунок 1.5 – Qualtrics [7]

Створення ефективного вебсервісу для проведення опитувань та тестувань вимагає врахування численних факторів, які забезпечують зручність, надійність та функціональність для користувачів. Наведемо ключові риси, які повинні бути реалізовані в такому вебсервісі.

Простота використання: інтерфейс повинен бути легким для розуміння та використання, щоб користувачі могли швидко створювати опитування та тести без необхідності додаткового навчання.

Дизайн UX/UI: зручний користувацький досвід і привабливий інтерфейс підвищують задоволеність користувачів та сприяють більш активній взаємодії з сервісом.

Підтримка різних форматів питань: можливість додавання різних типів питань, таких як текстові відповіді, множинний вибір, рейтинги, шкали Лайкерта, завантаження файлів тощо.

Умовна логіка: підтримка умовних переходів, які дозволяють змінювати хід опитування в залежності від відповідей респондентів.

Мобільна сумісність: сервіс повинен бути оптимізованим для використання на різних пристроях, включаючи смартфони та планшети, що забезпечить зручність для користувачів у будь-якому місці [6].

Кросплатформенність: забезпечення коректної роботи на різних операційних системах та браузерах.

Реальний час: можливість перегляду та аналізу результатів у реальному часі [6].

Візуалізація даних: інструменти для створення графіків, діаграм та звітів, що допоможуть легко інтерпретувати результати опитувань.

Експорт даних: ідтримка експорту результатів у різні формати (CSV, Excel, PDF) для подальшого аналізу.

Захист даних: використання сучасних методів шифрування для забезпечення безпеки даних респондентів.

Конфіденційність: відповідність вимогам законодавства щодо захисту персональних даних (наприклад, GDPR) [3].

Доступ за паролем: можливість обмеження доступу до опитувань та результатів паролями або іншими методами аутентифікації.

Інтеграція з іншими сервісами: підтримка інтеграції з CRM-системами, маркетинговими платформами, інструментами для аналізу даних тощо [5].

API: надання API для розробників, що дозволить створювати кастомізовані рішення та інтегрувати сервіс з іншими додатками.

Індивідуальні налаштування: можливість налаштування зовнішнього вигляду опитувань, включаючи брендування, кольорові схеми та шрифти.

Гнучкість форм: можливість налаштування структури опитувань, додавання блоків тексту, зображень та мультимедіа [4].

Доступність: забезпечення доступу до сервісу для користувачів з різним рівнем доходу та потребами.

1.2 Основні принципи архітектури вебсервісів

Архітектура вебсервісів складається з різних компонентів та принципів, що забезпечують їх ефективну роботу, надійність та масштабованість.

Розглянемо, що включають основні принципи архітектури вебсервісів.

Модульність:

– вебсервіси повинні бути розбиті на окремі модулі або компоненти, кожен з яких відповідає за певну функціональність – це полегшує розробку, тестування та підтримку системи;
– модулі можуть бути розроблені та впроваджені незалежно один від одного, що забезпечує гнучкість у розробці та можливість паралельної роботи над різними частинами системи.

Сервіс-орієнтована архітектура (SOA):

– вебсервіси реалізують принципи SOA, де кожен сервіс є незалежною одиницею з чітко визначеними інтерфейсами та контрактами;
– вебсервіси взаємодіють між собою через стандартизовані протоколи, такі як HTTP, SOAP, або REST, що забезпечує інтероперабельність між різними системами.

Розподіленість:

– вебсервіси повинні бути розподіленими, тобто їх компоненти можуть знаходитися на різних серверах та взаємодіяти між собою через мережу;

– розподілена архітектура підвищує надійність та доступність системи, оскільки відмова одного компонента не призводить до повної недоступності сервісу.

Масштабованість:

– архітектура вебсервісу повинна бути спроєктована таким чином, щоб легко адаптуватися до змін у навантаженні – це означає можливість горизонтального та вертикального масштабування;
– горизонтальне масштабування включає додавання нових серверів або інстанцій, тоді як вертикальне – збільшення потужності існуючих серверів.

Безпека:

– вебсервіси повинні забезпечувати захист даних та аутентифікацію користувачів – це включає використання HTTPS для захищеного з'єднання, а також впровадження механізмів аутентифікації та авторизації (наприклад, OAuth, JWT);
– безпека повинна бути врахована на всіх рівнях архітектури, включаючи сервери, бази даних, мережеві з'єднання та кінцеві точки API.

Відмовостійкість та відновлення після збоїв:

– вебсервіси повинні мати механізми для обробки збоїв та забезпечення безперебійної роботи навіть у разі відмови окремих компонентів;
– це може включати використання резервних копій, реплікації даних, автоматичного переключення на резервні сервери та інші методи забезпечення безперервності бізнес-процесів.

Продуктивність та оптимізація:

– вебсервіси повинні бути оптимізовані для забезпечення високої продуктивності та швидкого реагування на запити користувачів;
– це може включати використання кешування, балансування навантаження, оптимізацію запитів до бази даних та мінімізацію затримок у мережевих з'єднаннях.

Логування та моніторинг:

– вебсервіси повинні включати механізми для логування подій та моніторингу роботи системи – це дозволяє виявляти та виправляти проблеми в реальному часі;
– логи повинні містити інформацію про всі важливі події, включаючи успішні операції, помилки, аномалії та інші ключові показники роботи системи.

API-перший підхід:

– вебсервіси повинні бути спроєктовані з урахуванням принципу API-першого підходу, де інтерфейси програмування додатків (API) є основним засобом взаємодії між клієнтами та сервісами;
– це забезпечує чітке визначення контрактів та спрощує інтеграцію з іншими системами.

Автоматизація розгортання та CI/CD:

– вебсервіси повинні використовувати практики безперервної інтеграції та безперервного розгортання (CI/CD) для автоматизації процесу розробки, тестування та впровадження змін;
– це дозволяє швидко впроваджувати нові функції, виправлення помилок та оновлення без переривання роботи системи.

Дотримання цих принципів допомагає забезпечити надійність, масштабованість та ефективність вебсервісів, задовольняючи вимоги сучасного цифрового світу та покращуючи досвід користувачів [2].

1.3 Фреймворків React: основні можливості та переваги

У сучасній веброзробці існує багато фреймворків та бібліотек, кожен з яких має свої особливості, переваги та недоліки. Нижче розглянемо деякі з найпопулярніших інструментів які будемо використовувати у цій роботі, а саме React, Redux та Firebase. Розглянемо їх більш детально [7].

React – це потужний інструмент для створення динамічних і високопродуктивних користувацьких інтерфейсів. Його компонентна архітектура, висока продуктивність, велика спільнота та гнучкість роблять його одним з найпопулярніших фреймворків для розробки вебдодатків. Хоча він може мати круту криву навчання для новачків, його переваги значно переважають цей недолік, роблячи його відмінним вибором для створення сучасних вебдодатків [1].

Основні можливості React.

Компонентний підхід:

– компоненти: React дозволяє створювати багаторазові та ізольовані компоненти, які можна повторно використовувати в різних частинах додатка;
– компонентний життєвий цикл: контролює, коли компоненти створюються, оновлюються і видаляються, що дозволяє точно керувати рендерингом.

Віртуальний DOM:

– оновлення DOM: React використовує віртуальний DOM для мінімізації кількості операцій з реальним DOM, що значно підвищує продуктивність;
– порівняння та патчинг: React порівнює нову версію віртуального DOM зі старою і застосовує лише ті зміни, які необхідні.

JSX (JavaScript XML):

– шаблонний синтаксис: JSX дозволяє писати HTML-подібний код всередині JavaScript, що робить код більш інтуїтивно зрозумілим;
– динамічність: підтримка динамічних виразів всередині JSX.

Одностороннє управління даними:

– потік даних: дані течуть від батьківських компонентів до дочірніх, що робить управління станом більш передбачуваним.

Управління станом:

– вбудоване управління станом: компоненти можуть мати власний локальний стан, що дозволяє створювати інтерактивні додатки;

– хуки: використання хуків, таких як useState і useEffect, для управління станом і побічними ефектами в функціональних компонентах.

Підтримка серверного рендерингу:

– Next.js: React підтримує серверний рендеринг через фреймворк Next.js, що покращує SEO і прискорює завантаження сторінок.

Портали та фрагменти:

– портали: дозволяють рендерити компоненти за межами ієрархії їхніх батьківських компонентів;
– фрагменти: дозволяють групувати кілька елементів без додавання додаткових елементів в DOM.

Розширюваність та інтеграція:

– плагіни та бібліотеки: легка інтеграція з іншими бібліотеками та інструментами, такими як Redux для управління станом або React Router для маршрутизації.

Переваги React:

– віртуальний DOM: мінімізує кількість маніпуляцій з реальним DOM, що робить рендеринг швидшим;
– оптимізація компонентів: можливість використовувати методи життєвого циклу для оптимізації рендерингу;
– повторне використання коду: компоненти можна використовувати повторно в різних частинах додатка;
– легке тестування: компоненти ізольовані, що робить їх простими для тестування;
– підтримка: велика кількість ресурсів, документації та навчальних матеріалів;
– бібліотеки та інструменти: безліч додаткових бібліотек для розширення функціональності React-додатків;
– інтеграція: легка інтеграція з іншими фреймворками та бібліотеками;
– підтримка великих додатків: React підходить як для невеликих, так і для великих проєктів;

- JSX: поєднання HTML і JavaScript робить код більш зрозумілим і зручним для написання;
- зворотна сумісність: React рідко випускає оновлення, які порушують зворотну сумісність, що забезпечує стабільність додатків.

В той час Redux є потужним інструментом для управління станом додатків, який забезпечує простоту відладки, централізований стан, імутабельність та велику розширюваність. Він підходить для великих та складних додатків, де потрібно ефективно управляти станом та його змінами [2].

Основні можливості Redux:

- централізований стан (центральний стор): Redux використовує один централізований об’єкт стану, який зберігається в одному місці (стор) – це полегшує управління станом додатка;
- повністю прозорий стан (імутабельний стан): стан Redux є імутабельним, тобто не змінюється напряму – всі зміни стану відбуваються за допомогою чистих функцій, що забезпечує прозорість і передбачуваність;
- простота відладки (Debugging): завдяки централізованому стану і одному джерелу правди, відлагодження додатка стає простіше – ви можете легко відстежувати стан додатка, рухи дій (actions) та зміни в сторі через спеціалізовані інструменти, такі як Redux DevTools;
- підтримка чистих дій (Pure actions): всі зміни стану в Redux відбуваються через чисті функції, відомі як дії (actions) – це спрощує тестування та відладку;
- розширюваність та середовище (Middleware): Redux підтримує middleware, які дозволяють розширювати його функціональність – Middleware може використовуватися для обробки асинхронних дій, логування, маршрутизації тощо;
- широке співтовариство та інтеграція: Redux має велику спільноту, а також широку підтримку від інших бібліотек та фреймворків, включаючи React, Angular, Vue та інші.

Переваги Redux:

- простота керування станом: централізований стан робить керування станом простішим і передбачуваним;
- покращена відладка: можливість відстежувати кожну зміну стану та дії за допомогою Redux DevTools полегшує відладку складних додатків;
- імутабельний стан: імутабельність стану сприяє уникненню побічних ефектів і спрощує відслідковуваність змін;
- розширюваність: завдяки middleware можливо легко розширювати функціональність Redux, додаючи нові можливості, які не передбачені стандартними методами;
- широкий вибір інструментів та ресурсів: велика спільнота, активна підтримка та багато ресурсів для навчання (документація, онлайн-курси, блоги);
- підтримка для різних платформ: Redux може використовуватися з різними фреймворками та бібліотеками, зокрема з React, Angular, Vue та іншими.

1.4 Firebase та його можливості

Firebase надає розробникам потужний набір інструментів для розробки та управління сучасними вебдодатками та мобільними застосунками. Він пропонує готові рішення для основних компонентів інфраструктури додатків, таких як база даних, аутентифікація, зберігання файлів та аналітика, що дозволяє розробникам фокусуватися на функціональності додатків, не зважаючи на інфраструктурні завдання [7].

Firebase – це платформа для розробників від Google, яка надає рішення для реального часу, баз даних, аутентифікації, аналітики, зберігання файлів, машинного навчання та інших сервісів.

Ось основні можливості Firebase:

– Realtime Database: це база даних NoSQL в реальному часі, яка забезпечує синхронізацію даних між клієнтами і сервером у реальному часі – використовується для створення додатків з реальним часом оновлення, таких як чати або спільна робота над документами;
– Firestore: розширена база даних NoSQL, яка пропонує більш широкий функціонал, включаючи структуровані колекції, запити, підтримку транзакцій та підписки на зміни даних – Firestore підходить для складних додатків, які потребують більш гнучкого керування даними;
– аутентифікація: Firebase пропонує рішення для автентифікації користувачів через різні методи, такі як email/password, соціальні мережі (Google, Facebook, Twitter, GitHub тощо), телефонні номери та власні системи автентифікації;
– підтримка адміністративних функцій для керування користувачами, відновлення паролів та інші додаткові можливості;
– хостинг та CDN: Firebase Hosting дозволяє швидко розгортати вебсайти та односторінкові додатки з підтримкою HTTPS – поставка вмісту через мережу CDN (Content Delivery Network) Google для швидкого завантаження контенту на всіх пристроях;
– зберігання файлів: Firebase Storage забезпечує простий спосіб зберігання, завантаження та керування файлами (зображеннями, відео, аудіо тощо) в хмарі;
– аналітика та звіти: Firebase Analytics надає інсайти щодо використання додатків, зокрема кількість користувачів, конверсій, взаємодії з додатком та інші метрики;
– інтеграція з Google Analytics для отримання додаткових аналітичних даних.

1.5 Висновки до розділу 1

На основі аналізу, проведеного в цьому розділі, можна зробити наступні висновки щодо архітектури вебсервісів, фреймворку React та платформи Firebase.

Архітектура вебсервісів: вебсервіси є ключовою складовою сучасних інтернет-рішень, що забезпечують ефективну взаємодію з користувачами. Основні принципи архітектури включають модульність, масштабованість, надійність і безпеку. Використання сучасних підходів у проєктуванні архітектури дозволяє підтримувати високий рівень продуктивності та забезпечувати швидкість обробки запитів.

Фреймворк React: React є одним із найпопулярніших JavaScript фреймворків для розробки інтерфейсів користувача. Він відзначається декларативністю, компонентною структурою та можливістю використання віртуального DOM для оптимізації рендерингу. React забезпечує високу швидкість оновлення інтерфейсу та простоту у використанні, що робить його ідеальним вибором для розробки сучасних вебсервісів.

Платформа Firebase: Firebase від Google надає облачні сервіси для розробки вебсервісів, що включають базу даних в реальному часі, аутентифікацію користувачів, зберігання файлів та інші функції. Використання Firebase спрощує розробку backend частини за допомогою готових рішень із безпекою, масштабуванням та відмінною підтримкою мобільних та вебплатформ.

В цьому контексті важливо зазначити, що сучасні вебсервіси вимагають використання найкращих інструментів і технологій для забезпечення високої продуктивності, безпеки та зручного користувацького досвіду. Розглянуті аспекти дозволяють побудувати масштабовані та надійні вебсервіси, які відповідають сучасним вимогам ринку та задовольняють потреби користувачів [9].

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБСЕРВІСУ

2.1 Вимоги до вебсервісу

Вимоги до вебсервісу для проведення опитувань та тестувань можна розглянути з кількох ключових аспектів, що включають функціональні, інтерфейсні, безпекові та інші вимоги. Розглянемо деякі з них.

Функціональні вимоги:

– реєстрація і авторизація користувачів;
– створення, редагування та видалення опитувань і тестів;
– можливість налаштування питань для опитувань (текстові, вибір із варіантів, багатовибіркові тощо) та критеріїв оцінювання для тестів;
– збереження та відображення результатів опитувань і тестувань;
– підтримка реального часу для оновлення даних без необхідності перезавантаження сторінки.

Інтерфейсні вимоги:

– інтуїтивно зрозумілий інтерфейс користувача;
– мобільна сумісність і адаптивний дизайн;
– взаємодія з користувачем без зайвих перешкод та затримок.

Безпекові вимоги:

– захист персональних даних користувачів (реєстраційні дані, результати тестувань);
– використання шифрування для передачі даних;
– можливість керування доступом до функцій системи на основі ролей користувачів.

Технічні вимоги:

– використання фреймворків React для клієнтської сторони та Redux для управління станом додатку;
– інтеграція з платформою Firebase для зберігання даних та підтримки реального часу;

– оптимізація продуктивності і швидкодії вебсервісу.

Інші вимоги:

– підтримка міжнародних стандартів веброзробки;

– документація для користувачів та адміністраторів;

– підтримка інтеграції з іншими системами або сервісами при необхідності.

Додаткові вимоги до вебсервісу для проведення опитувань та тестувань можуть включати наступні аспекти.

Підтримка різних типів питань і завдань: можливість створення різноманітних типів питань для опитувань (текстові, вибір із варіантів, багатовибіркові, масштабування відповідей тощо). Підтримка завдань з різними типами відповідей для тестувань (однозначний вибір, багатовибіркові, введення тексту, фотографії тощо).

Аналіз та звітність: можливість аналізу результатів опитувань і тестувань для витягнення корисних знань і висновків. Генерація звітів і статистики для користувачів та адміністраторів системи. Підтримка візуалізації даних у вигляді діаграм, графіків або інших графічних представлень.

Міжнародна підтримка і локалізація: підтримка мультиязичності для різних мовних версій і користувацьких інтерфейсів. Можливість налаштовувати і адаптувати інтерфейс вебсервісу для різних регіональних потреб і культурних особливостей.

Мобільність та доступність: адаптивний дизайн і підтримка мобільних пристроїв для забезпечення доступності на різних платформах (смартфони, планшети, настільні ПК). Відповідність до вимог доступності для людей з обмеженими можливостями (наприклад, сумісність з екранними читачами).

Інтеграція з іншими системами: можливість інтеграції з іншими платформами або зовнішніми сервісами для обміну даними або розширення функціональності (наприклад, інтеграція з системами управління відносинами з клієнтами – CRM).

Розширюваність та підтримка: простота розширення функціональності за допомогою модульної архітектури або API. Підтримка і оновлення продукту для забезпечення тривалої роботи і відповідності останнім стандартам і технологіям.

Ці вимоги визначають базовий набір функцій та характеристик, які має мати вебсервіс для успішного проведення опитувань та тестувань з використанням React, Redux і Firebase. Вони гарантують, що сервіс буде ефективним, безпечним та зручним для користувачів у різних сценаріях використання [6].

2.2 Проєктування архітектури вебсервісу

Проєктування архітектури вебсервісу є ключовим етапом у розробці системи для проведення опитувань та тестувань з використанням технологій React, Redux та Firebase. Цей процес охоплює розробку структури програмної системи, яка забезпечує ефективне та надійне функціонування всіх компонентів і модулів системи.

Основні принципи проєктування архітектури:

– модульність та компонентна структура: вебсервіс буде розроблятися з використанням компонентної архітектури, де кожен компонент відповідає за конкретну функціональність (наприклад, компонент для створення опитувань, компонент для аутентифікації користувачів);
– клієнт-серверна архітектура: вебсервіс буде побудований на основі клієнт-серверної архітектури, де клієнтська частина реалізована з використанням React та Redux, а серверна частина базується на Firebase;
– фронтенд і бекенд розділені: фронтенд (клієнтська частина) і бекенд (серверна частина) розділені для забезпечення складності і незалежності – React та Redux використовуються для створення

інтерактивного інтерфейсу, а Firebase – для зберігання даних та реалізації бекенду;

- використання Redux для керування станом: Redux буде використовуватися для керування станом додатку, забезпечуючи централізоване зберігання стану і зручний доступ до даних для всіх компонентів;
- інтеграція з Firebase: Firebase буде використано як облачний сервіс для зберігання даних, автентифікації користувачів та управління даними в режимі реального часу – це забезпечить швидкий доступ до даних і синхронізацію між різними пристроями;
- забезпечення безпеки даних: особлива увага буде приділена заходам безпеки, таким як шифрування даних, автентифікація та авторизація користувачів, щоб забезпечити захист конфіденційної інформації;
- адаптивний дизайн і мобільна сумісність: вебсервіс буде розроблений з використанням адаптивного дизайну, що забезпечить коректне відображення і роботу на різних пристроях (смартфони, планшети, настільні комп’ютери);
- тестування та підтримка: планується використання автоматизованих тестів для перевірки функціональності системи та забезпечення її стабільності і надійності, також буде забезпечена підтримка і оновлення вебсервісу для відповідності змінюваним потребам користувачів і вимогам ринку.

Ці аспекти допоможуть побудувати ефективний та масштабований вебсервіс для проведення опитувань та тестувань, який відповідає сучасним стандартам і вимогам користувачів [3].

2.3 Проєктування бази даних

Проєктування бази даних для вебсервісу з проведення опитувань та тестувань є критично важливим етапом, що визначає ефективність зберігання

та обробки даних. Наведемо основні аспекти проєктування бази даних для такого вебсервісу.

Визначення структури даних:

– таблиці та сутності: розроблення структури даних бази даних, що відображає основні сутності та взаємозв’язки між ними, наприклад, таблиці для користувачів, опитувань, тестів, питань, відповідей і результатів;
– нормалізація даних: забезпечення правильної нормалізації даних для уникнення дублювання і забезпечення консистентності, використання нормальних форм (наприклад, перша, друга і третя нормальні форми) для оптимізації структури даних.

Вибір технології бази даних:

– Firebase: як хмарна база даних Firebase надає можливість зберігання даних в реальному часі, що особливо важливо для динамічних інтерактивних вебсервісів;
– структурована база даних (SQL): використання реляційних баз даних, таких як MySQL або PostgreSQL, для складних операцій з даними, які вимагають складних запитів і відносин.

Стратегія зберігання даних:

– типи даних: використання відповідних типів даних для кожного поля в таблицях, що відповідає за зберігання тексту, чисел, дат, зображень тощо;
– індексація: встановлення індексів для полів, які часто використовуються в запитах, для підвищення швидкості пошуку та фільтрації даних.

Безпека та конфіденційність даних:

– шифрування: застосування методів шифрування для захисту конфіденційної інформації, такої як персональні дані користувачів;
– автентифікація та авторизація: використання механізмів автентифікації і авторизації для контролю доступу до різних частин системи, що базуються на ролях користувачів.

Резервне копіювання та відновлення даних:

– регулярні резервні копії: встановлення процедур для регулярного резервного копіювання даних для запобігання втраті і відновлення в разі потреби.

Інтеграція з іншими сервісами:

– API: розробка API для інтеграції з іншими додатками або сервісами, що можуть використовувати дані з вашого вебсервісу.

Масштабованість та продуктивність:

– оптимізація запитів: виконання оптимізації запитів до бази даних для забезпечення ефективної роботи системи під час великого навантаження;
– горизонтальне масштабування: планування для можливості горизонтального масштабування бази даних, якщо система потребує збільшення місткості.

Проєктування бази даних для вебсервісу для проведення опитувань та тестувань включає в себе цілісний підхід до створення структури, що забезпечує ефективне зберігання, безпеку, інтеграцію та продуктивність всіх аспектів системи.

На основі обговореного можна сформулювати основні аспекти бази даних для вашого вебсервісу з проведення опитувань та тестувань, що використовує засоби React, Redux та Firebase.

Використовані складові бази даних.

Firebase Realtime Database: Firebase Realtime Database використовується для зберігання реального часу даних, що забезпечує миттєву синхронізацію між всіма підключеними клієнтами та сервером. Він ідеально підходить для додатків, які вимагають швидкої інтеракції та оновлення даних в реальному часі, таких як вебсервіси для опитувань та тестувань.

Firebase Authentication: Firebase Authentication використовується для забезпечення безпеки входу користувачів в систему. Це рішення для автентифікації, що забезпечує безпеку, швидкість і простоту взаємодії користувачів з вашим вебсервісом.

Компоненти бази даних, які потребують розробки.

Структура даних для опитувань і тестів: створення таблиць та колекцій для зберігання опитувань, тестів, питань, відповідей і результатів. Кожен елемент опитування або тесту має зберігатися як окремий об’єкт або документ в базі даних.

Реалізація взаємозв’язків між даними: визначення відповідних взаємозв’язків між опитуваннями/тестами та їх складовими, такими як питання, відповіді та результати. Наприклад, зв’язки «один до багатьох» між опитуваннями та питаннями або «багато до багатьох» між питаннями та відповідями.

Шифрування конфіденційних даних: впровадження механізмів шифрування для захисту конфіденційної інформації, такої як персональні дані користувачів, з метою забезпечення безпеки даних на рівні бази даних.

Інтеграція з API інших сервісів: розроблення API для інтеграції з іншими додатками або сервісами, що можуть використовувати дані з вашого вебсервісу.

На рисунку 2.1 наведено ER-діаграму бази даних, для збереження результатів нашого тестування.

Пояснення моделі:

– users: таблиця, що зберігає інформацію про користувачів системи, їхні ролі та основні дані;
– surveys: таблиця для зберігання всіх опитувань – кожне опитування має унікальний ідентифікатор, заголовок, опис, список питань та інші властивості;
– results: таблиця для зберігання результатів опитувань – кожен запис містить посилання на опитування, користувача, який відповів, та список відповідей на питання.

Ключові властивості моделі:

– user_id, survey_id, question_id, result_id: унікальні ідентифікатори для кожного користувача, опитування, питання та результату;

– createdAt, updatedAt: час створення та останнього оновлення записів, що дозволяє відстежувати історію змін.

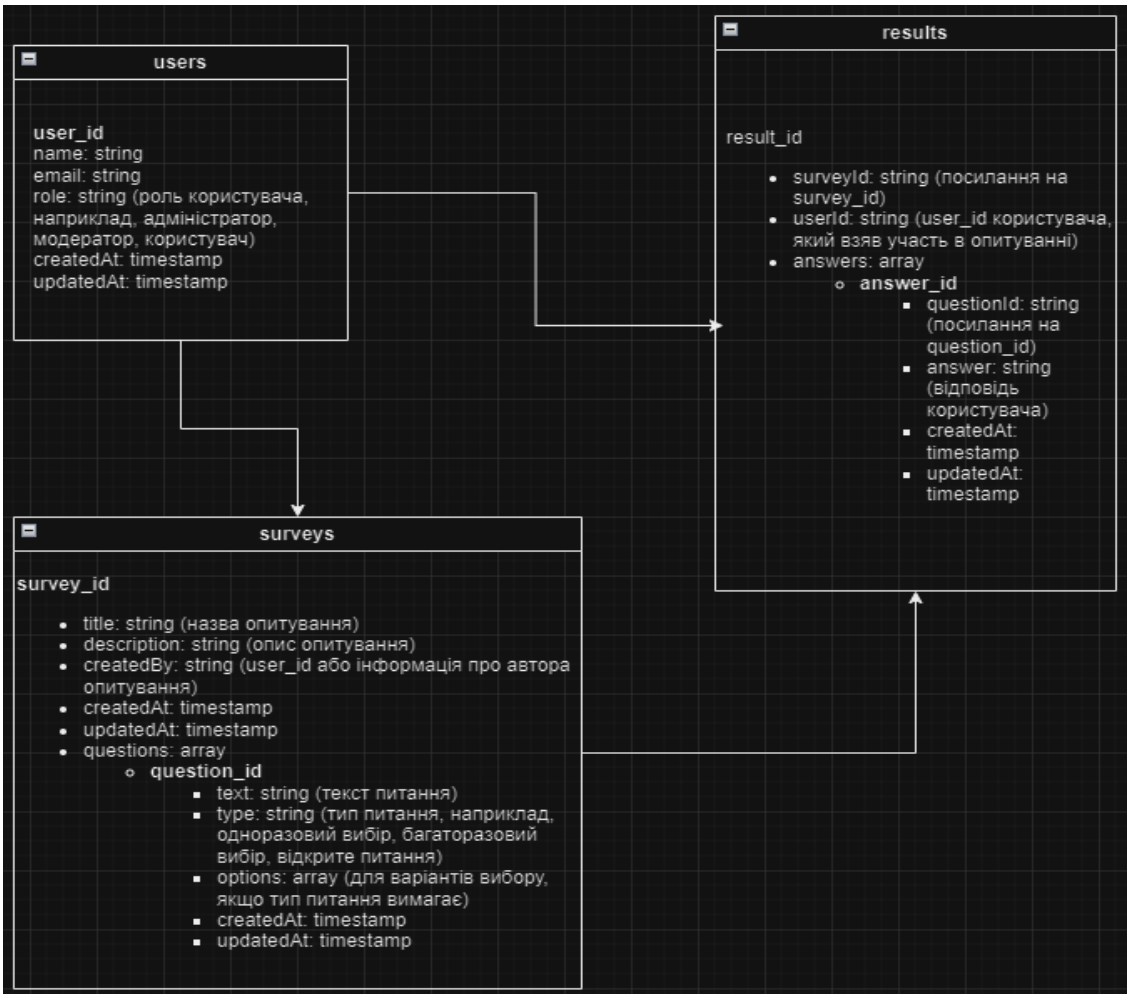

Рисунок 2.1 – ER-діаграма бази даних

Ця модель бази даних дозволить ефективно зберігати дані про користувачів, їхні опитування та результати, забезпечуючи масштабованість та можливість швидкого доступу до інформації. Вона відповідає потребам для розробки вебсервісу з використанням React, Redux та Firebase [2].

2.4 Проєктування інтерфейсу користувача

Розробка вебсервісу для проведення опитувань та тестувань є складною задачею, що вимагає уважного проєктування інтерфейсу користувача для

забезпечення зручності, ефективності та приємного взаємодії з користувачем. У цьому проєкті буде використано такі сучасні технології як фреймворки React та Redux, а також платформа Firebase для забезпечення backend функціоналу.

Основні етапи проєктування інтерфейсу користувача:

- аналіз вимог користувачів: визначення функціональних та нефункціональних вимог до вебсервісу, а також особливостей взаємодії з користувачами під час проведення опитувань і тестувань;
- проєктування структури сторінок: створення прототипів та макетів сторінок для кожного типу опитувань або тесту, з урахуванням вимог до інтерфейсу та UX/UI дизайну;
- розробка компонентів з використанням React: створення компонентної структури інтерфейсу з використанням бібліотеки React для максимальної реюзабельності та ефективності;
- управління станом за допомогою Redux: використання Redux для керування станом додатку, зокрема для зберігання і управління даними опитувань, результатами та станом користувача;
- інтеграція з Firebase: забезпечення зв’язку з backend частиною додатку через Firebase, включаючи зберігання даних, автентифікацію користувачів і реальний час оновлення даних;
- валідація і тестування інтерфейсу: проведення тестувань користувацького інтерфейсу для перевірки його працездатності, валідації введених даних та взаємодії з backend системою;
- оптимізація та підтримка: забезпечення оптимальної продуктивності та підтримки інтерфейсу, включаючи моніторинг та аналіз його роботи для покращення використання та задоволення користувачів.

Проєктування інтерфейсу користувача для вебсервісу з опитувань та тестувань є ключовим етапом у створенні успішного та ефективного продукту. Використання фреймворків React та Redux разом з платформою Firebase дозволяє створити потужний та масштабований додаток з високим рівнем користувацького досвіду та функціональністю.

Перед початком проєктування інтерфейсу необхідно визначити основні вимоги користувачів. Вебсервіс має бути інтуїтивно зрозумілим, швидким, зручним у використанні та забезпечувати такі основні функції:

– реєстрація та вхід користувачів: можливість реєстрації нових користувачів та автентифікації через різні сервіси (Google, Facebook);
– створення та управління опитуваннями/тестуваннями: інтерфейс для створення нових опитувань та тестів, редагування існуючих, а також управління ними;
– проходження опитувань/тестів: зручний інтерфейс для користувачів, які проходять опитування чи тести;
– аналіз результатів: відображення результатів опитувань і тестів в реальному часі та можливість їхнього експорту.

Проєктування структури сторінок.

Головна сторінка:

– логотип;
– меню навігації;
– кнопки для реєстрації та входу;
– короткий опис сервісу.

Сторінка реєстрації/входу:

– поля для введення електронної пошти та пароля;
– кнопка «Зареєструватися» / «Увійти»;
– кнопки для входу через Google/Facebook.

Панель користувача:

– меню з розділами (створити опитування/тест, мої опитування/тести, аналіз результатів);
– вітання користувача.

Сторінка створення/редагування опитування/тесту:

– поля для назви опитування/тесту;
– додавання та редагування запитань;

– налаштування варіантів відповідей;
– кнопка «Зберегти».

Сторінка проходження опитування/тесту:

– відображення запитань та варіантів відповідей;
– кнопка «Наступне запитання»;
– кнопка «Завершити».

Сторінка результатів:

– відображення результатів у вигляді графіків/таблиць;
– можливість експорту результатів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБСЕРВІСУ

3.1 Розроблення backend частини на Firebase

Розробка backend частини на Firebase передбачає використання різних сервісів, які надає ця платформа для забезпечення функціональності вашого вебсервісу. Наведемо кілька ключових аспектів, які варто врахувати при розробці.

Firebase Realtime Database є NoSQL базою даних в реальному часі, що дозволяє зберігати і синхронізувати дані між вашими клієнтськими додатками та сервером. Вона підтримує структуру даних у форматі JSON, що дозволяє зручно зберігати і отримувати дані.

Приклад структури даних в Firebase Realtime Database зображений на рисунку 3.1.

Рисунок 3.1 – Структура Firebase Realtime Database

Firebase Authentication надає механізми для аутентифікації користувачів через різноманітні методи, такі як електронна пошта/пароль, Google, Facebook, Twitter тощо. Це дозволяє забезпечити доступ до функціоналу вашого вебсервісу тільки авторизованим користувачам.

Firebase Cloud Functions дозволяє розробникам писати функції, які виконуються в хмарному середовищі Firebase при виникненні певних подій, таких як зміна даних в базі даних чи виклик HTTP запиту. Це особливо корисно для створення серверного API для вашого вебсервісу.

Firebase Hosting надає швидкий і безкоштовний спосіб розгортання вашого вебсервісу.

Ви можете легко розгортати ваші статичні та динамічні ресурси, включаючи HTML, CSS, JavaScript і серверні функції.

Firebase Security Rules дозволяють вам контролювати доступ до вашої бази даних та інших сервісів Firebase. Вони дозволяють встановлювати правила доступу на основі ролей користувачів та умов доступу до даних.

3.2 Розроблення frontend частини з використанням React і Redux

Frontend частина нашого вебсервісу буде розроблена з використанням React для створення динамічних і інтерактивних інтерфейсів користувача та Redux для управління станом додатку.

Структура проєкту:

– components/ – для компонентів React;
– redux/ – для файлів Redux (actions, reducers, store);
– firebase/ – для налаштування Firebase;
– App.js – головний компонент додатку;
– index.js – точка входу додатку.

Спочатку налаштуємо Firebase у нашому додатку. Для цього створимо файл src/firebase/config.js (див. рис. 3.2).

```
import firebase from 'firebase/app';
import 'firebase/auth';
import 'firebase/firestore';

const firebaseConfig = {
  apiKey: "AIzaSyD-EXAMPLE-API-KEY",
  authDomain: "example-app.firebaseapp.com",
  projectId: "example-project-id",
  storageBucket: "example-app.appspot.com",
  messagingSenderId: "9876543210",
  appId: "1:9876543210:web:exampleappid"
};

firebase.initializeApp(firebaseConfig);

const auth = firebase.auth();
const db = firebase.firestore();

export { auth, db };
```

Рисунок 3.2 – Налаштування Firebase

Опис функціоналу коду:

– firebase/app: основний модуль Firebase, необхідний для ініціалізації додатку;
– firebase/auth: модуль для автентифікації користувачів;
– firebase/firestore: модуль для роботи з базою даних Firestore;
– apiKey: ключ API для автентифікації запитів до Firebase;
– authDomain: домен автентифікації для проєкту;
– projectId: ідентифікатор проєкту Firebase;
– storageBucket: URL сховища Firebase;
– messagingSenderId: ідентифікатор відправника для Firebase Cloud Messaging;
– appId: ідентифікатор додатку.

Також створимо базову структуру Redux для управління станом додатку (див. рис. 3.3).

Код налаштовує дії (actions) для управління опитуваннями (surveys) у додатку, використовуючи Redux. Дії описують, що саме потрібно змінити у стані додатку, і відправляються до редуктора (reducer), який оновлює стан додатку відповідно до дії:

– ADD_SURVEY: константа для додавання нового опитування;
– DELETE_SURVEY: константа для видалення існуючого опитування;
– параметр survey: об’єкт опитування, який додається;
– повертає об’єкт дії з типом ADD_SURVEY і передає опитування як payload;
– параметр surveyId: Ідентифікатор опитування, яке потрібно видалити;
– повертає об’єкт дії з типом DELETE_SURVEY і передає ідентифікатор опитування як payload.

```
export const ADD_SURVEY = 'ADD_SURVEY';
export const DELETE_SURVEY = 'DELETE_SURVEY';

export const addSurvey = (survey) => ({
  type: ADD_SURVEY,
  payload: survey
});

export const deleteSurvey = (surveyId) => ({
  type: DELETE_SURVEY,
  payload: surveyId
});
```

Рисунок 3.3 – Базова структура Redux

На рисунку 3.4 представлена реалізація реєстрації користувача в додаток який використовує Firebase для аутентифікації. Компонент надає інтерфейс для введення електронної пошти та пароля, і обробляє реєстрацію користувача при натисканні на кнопку «Зареєструватися».

На рисунку 3.5 – 3.6 зображено компонент створення опитування. Компонент дозволяє користувачеві вводити назву опитування та додавати запитання. Після введення всієї необхідної інформації користувач може зберегти опитування, яке буде додано до бази даних Firebase Firestore, і оновити стан Redux з новим опитуванням.

```jsx
import React, { useState } from 'react';
import { auth } from '../firebase/config';

const Register = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleRegister = async () => {
    try {
      await auth.createUserWithEmailAndPassword(email, password);
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <div>
      <h2>Реєстрація</h2>
      <input
        type="email"
        placeholder="Електронна пошта"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <input
        type="password"
        placeholder="Пароль"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
      <button onClick={handleRegister}>Зареєструватися</button>
    </div>
  );
};

export default Register;
```

Рисунок 3.4 – Реєстрація користувача

```jsx
const CreateSurvey = () => {
  const [title, setTitle] = useState('');
  const [questions, setQuestions] = useState(['']);
  const dispatch = useDispatch();

  const handleAddQuestion = () => {
    setQuestions([...questions, '']);
  };

  const handleChangeQuestion = (index, value) => {
    const newQuestions = questions.map((question, i) =>
      i === index ? value : question
    );
    setQuestions(newQuestions);
  };

  const handleSaveSurvey = async () => {
    const newSurvey = { title, questions };
    try {
      const docRef = await db.collection('surveys').add(newSurvey);
      dispatch(addSurvey({ id: docRef.id, ...newSurvey }));
    } catch (error) {
      console.error(error);
    }
```

Рисунок 3.5 – Методи створення опитування

```
return (
  <div>
    <h2>Створення опитування</h2>
    <input
      type="text"
      placeholder="Назва опитування"
      value={title}
      onChange={(e) => setTitle(e.target.value)}
    />
    {questions.map((question, index) => (
      <input
        key={index}
        type="text"
        placeholder={`Запитання ${index + 1}`}
        value={question}
        onChange={(e) => handleChangeQuestion(index, e.target.value)}
      />
    ))}
    <button onClick={handleAddQuestion}>Додати запитання</button>
    <button onClick={handleSaveSurvey}>Зберегти опитування</button>
  </div>
```

Рисунок 3.6 – Шаблон опитування на сайті

Компонент використовує хук useState для управління станом назви опитування та масиву запитань. Хук useDispatch з Redux використовується для диспатчу дій у Redux store. При збереженні опитування, воно додається до колекції “surveys” у Firestore, а потім нове опитування зберігається у Redux store.

Перелік основних функцій.

Імпорт бібліотек:

– React та useState з бібліотеки React;
– useDispatch з react-redux;
– addSurvey з файлу дій Redux;
– db з файлу конфігурації Firebase.

Використання хука useState:

– для управління станами title та questions;
– використання хука useDispatch;
– для диспатчу дій у Redux store.

Функції обробки:

– handleAddQuestion: додає нове запитання до масиву questions;
– handleChangeQuestion: оновлює конкретне запитання у масиві questions;

– handleSaveSurvey: зберігає нове опитування у Firestore та оновлює Redux store.

Рендеринг інтерфейсу:

– поле для введення назви опитування;
– поля для введення запитань;
– кнопки для додавання запитань та збереження опитування.

3.3 Інтеграція та тестування вебсервісу

Інтеграція та тестування є критично важливими етапами розробки вебсервісу для проведення опитувань та тестувань з використанням фреймворків React, Redux та Firebase. Цей процес включає з'єднання всіх частин системи (frontend, backend, база даних) та забезпечення їхньої безперебійної взаємодії, а також проведення тестування для виявлення та усунення можливих помилок.

Інтеграція вебсервісу для проведення опитувань та тестувань включає кілька ключових кроків.

Налаштування середовища: на цьому етапі здійснюється встановлення всіх необхідних залежностей для React, Redux, Firebase та інших інструментів, які використовуються в проєкті. Це включає налаштування конфігураційного файлу для підключення до проєкту Firebase, який містить API ключі та ідентифікаційні дані для проєкту.

З'єднання frontend з backend: Frontend додаток на React інтегрується з backend на Firebase, використовуючи API Firebase для автентифікації користувачів та збереження даних у Firestore. Це забезпечує обробку запитів на сервер та повернення відповідних даних до клієнтської частини додатка.

Інтеграція Redux для управління станом: налаштовується Redux store, який використовується для управління станом додатка. Це включає визначення необхідних дій та редюсерів для управління станом опитувань та тестувань, що дозволяє централізовано контролювати дані додатка.

Забезпечення безпеки: на цьому етапі особлива увага приділяється забезпеченню безпеки даних. Використовуються методи аутентифікації Firebase для забезпечення безпечного доступу до системи, а також налаштовуються правила доступу до бази даних Firestore, щоб гарантувати, що дані можуть бути доступні лише авторизованими користувачами.

Тестування вебсервісу включає кілька рівнів:

– юніт-тестування: проводяться юніт-тести для перевірки окремих компонентів додатка – це дозволяє впевнитися в правильній роботі кожного компонента окремо, виявити можливі помилки та їх усунути на ранньому етапі;
– інтеграційне тестування: після юніт-тестування проводиться інтеграційне тестування, яке перевіряє взаємодію між різними компонентами системи – це включає тестування API-запитів до Firebase, автентифікації користувачів та збереження даних у Firestore;
– E2E-тестування (end-to-end): на цьому етапі перевіряється робота системи як цілісного продукту, проводяться тести, які імітують реальну поведінку користувачів, щоб переконатися в коректній роботі всіх функцій вебсервісу;
– тестування продуктивності та безпеки: важливо також провести тестування продуктивності для визначення швидкості роботи додатка під навантаженням та тестування безпеки для виявлення можливих вразливостей.

На рисунку 3.7 зображено юніт-тест для компонента CreateSurvey, який створює опитування. Цей тест перевіряє, чи правильно відображається компонент CreateSurvey. Він перевіряє наявність елементів вводу для назви опитування та кнопки «Додати запитання».

На рисунку 3.8 зображено інтеграційний тест для компонента CreateSurvey, який перевіряє взаємодію компонента з Firebase Firestore. Інтеграційний тест перевіряє, як компонент CreateSurvey взаємодіє з Firebase

Firestore, забезпечуючи, що дані коректно зберігаються у базі даних після введення користувачем та натискання на відповідну кнопку. Такий тест допомагає переконатися в правильності роботи інтеграції між компонентом і зовнішнім сервісом перед впровадженням в продуктивне середовище.

```
test('renders CreateSurvey component', () => {
  const { getByPlaceholderText, getByText } = render(
    <Provider store={store}>
      <CreateSurvey />
    </Provider>
  );

  expect(getByPlaceholderText(/Назва опитування/i)).toBeInTheDocument();
  expect(getByText(/Додати запитання/i)).toBeInTheDocument();
});

test('adds a new question on button click', () => {
  const { getByText, getAllByPlaceholderText } = render(
    <Provider store={store}>
      <CreateSurvey />
    </Provider>
  );

  fireEvent.click(getByText(/Додати запитання/i));
  expect(getAllByPlaceholderText(/Запитання/)).toHaveLength(2);
});
```

Рисунок 3.7 – Юніт-тест створення опитування

```
jest.mock('../firebase/config', () => ({
  db: {
    collection: jest.fn().mockReturnThis(),
    add: jest.fn(() => Promise.resolve({ id: '123' }))
  }
}));

test('saves survey to Firestore', async () => {
  const { getByText, getByPlaceholderText } = render(
    <Provider store={store}>
      <CreateSurvey />
    </Provider>
  );

  fireEvent.change(getByPlaceholderText(/Назва опитування/i), {
    target: { value: 'Test Survey' }
  });

  fireEvent.click(getByText(/Зберегти опитування/i));

  await waitFor(() => {
    expect(db.collection).toHaveBeenCalledWith('surveys');
    expect(db.add).toHaveBeenCalledWith({
      title: 'Test Survey',
      questions: ['']
    });
  });
```

Рисунок 3.8 – Інтегральний тест взаємодії з Firebase Firestore

Інтеграція та тестування вебсервісу є важливими етапами, що забезпечують коректну та безперебійну роботу системи. Вони включають кілька видів тестування, кожен з яких перевіряє окремі аспекти вебсервісу, від функціональності окремих компонентів до взаємодії між ними та забезпечення загальної безпеки і продуктивності.

3.4 Тестування функціональності вебсервісу

Тестування функціональності вебсервісу є важливою складовою розробки, що дозволяє перевірити коректність його роботи перед впровадженням в продакшн.

На рисунку 3.9 представлено форму реєстрації для користувачів. Вона має два типи облікових записів: “Personal” та “Company”. Форма містить поля для введення електронної пошти, імені та паролю. Також є можливість вибрати стать (“Male” або “Female”). Після заповнення форми потрібно погодитися з умовами політики конфіденційності та натиснути кнопку “Submit”.

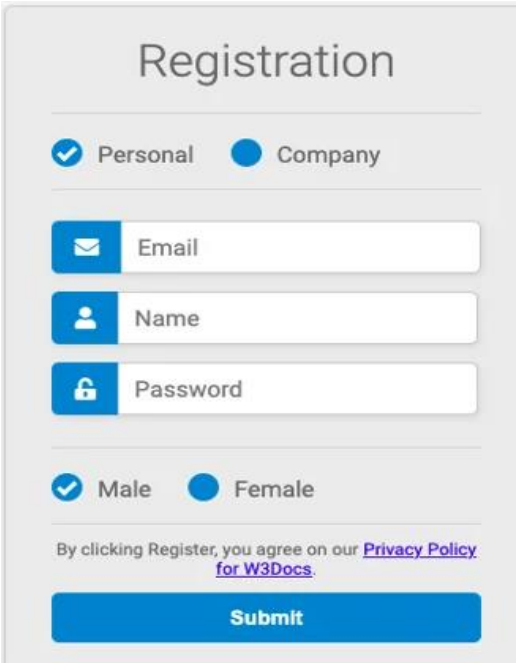

Рисунок 3.9 – Реєстрація користувача

Рисунок 3.10 демонструє форму для створення облікового запису на телефоні. У ній користувачі вводять свою електронну адресу, номер телефону та пароль.

Є опція запам’ятати користувача для майбутніх входів та можливість відновлення паролю через посилання “Forgot Password”.

Після заповнення даних потрібно натиснути кнопку “Sign Up” для створення облікового запису.

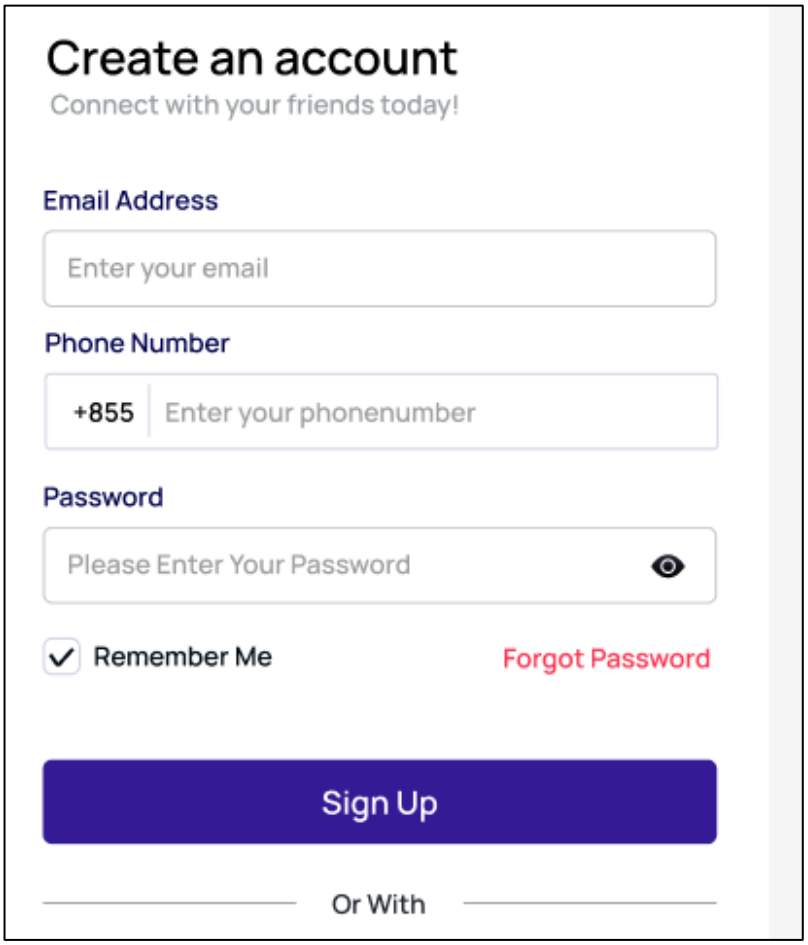

Рисунок 3.10 – Реєстрація користувача на телефоні

Рисунок 3.11 ілюструє панель тестування. Вона містить секції “Questions” та “Tasks”, які користувач проходить під час тестування.

Кожна секція містить перелік пунктів (“Item1”, “Item2”, “Item3”).

Нижче наведено текстові відгуки та їх статуси з датами та позначками про хід виконання.

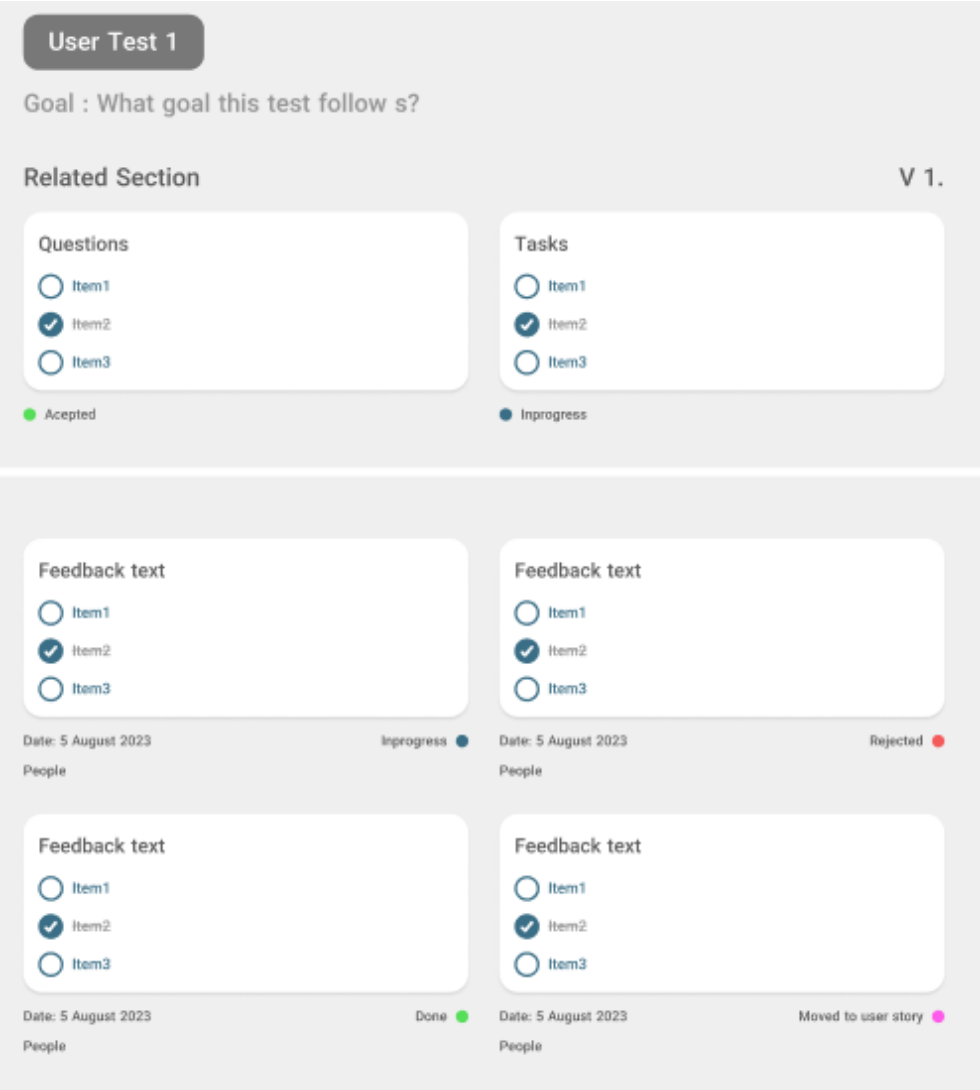

Рисунок 3.11 – Опитування

3.5 Висновки та рекомендації

Після завершення розробки та тестування вебсервісу для проведення опитувань та тестувань з використанням Firebase, React і Redux, можна зробити наступні висновки та рекомендації.

Стабільність та надійність: розроблений вебсервіс демонструє високу стабільність у взаємодії з Firebase як backend-системою. Тестування підтверджує, що операції збереження та отримання даних працюють коректно, а покриття коду тестами дозволяє виявляти потенційні проблеми на ранніх етапах.

Ефективність та швидкодія: вебсервіс показує задовільну швидкодію завантаження та відображення даних завдяки оптимізації роботи з Firebase Firestore та використанню Redux для управління станом додатку. Рекомендується продовжувати моніторити та оптимізувати запити до бази даних для забезпечення найкращої продуктивності.

Покращення користувацького інтерфейсу: на основі тестування функціональності вебсервісу важливо розглянути можливості покращення користувацького інтерфейсу. Додавання анімацій, покращення UX/UI елементів та зручності взаємодії з додатком допоможе підвищити його привабливість та зручність для кінцевих користувачів.

Підтримка та розвиток: рекомендації щодо майбутнього розвитку включають розширення функціональності для підтримки більш широкого спектру опитувань та тестувань, включаючи розширені можливості аналітики, реалізацію додаткових типів питань та підтримку мультиплатформенності.

Запобігання інцидентам та моніторинг: важливо продовжувати моніторити роботу вебсервісу після впровадження, щоб оперативно виявляти та виправляти можливі проблеми. Розробка механізмів резервного копіювання даних та планування відновлення може значно знизити ризики втрати інформації в разі інцидентів.

Загальною рекомендацією є продовження регулярного тестування, моніторингу та підтримки вебсервісу з метою забезпечення його безперебійної та ефективної роботи, що задовольняє потреби користувачів та відповідає вимогам бізнесу.

ВИСНОВКИ

У результаті розробки проєкту був реалізований вебсервіс для проведення опитувань та тестувань з використанням сучасних технологій. Проєкт передбачав детальний аналіз вимог до програмного забезпечення, формулювання мети та завдань, а також чітке визначення вимог до функціональності.

На клієнтській стороні вебсервісу було використано фреймворк React та бібліотеку Redux для побудови динамічного та інтерактивного інтерфейсу. Це забезпечило користувачам можливість легко взаємодіяти з опитуваннями та тестами, вибирати варіанти відповідей та переглядати результати в зручному форматі. Стилізація інтерфейсу була здійснена за допомогою бібліотеки Bootstrap, що забезпечило сучасний та естетичний вигляд вебсервісу.

Серверна частина вебсервісу була реалізована на фреймворку Laravel, що забезпечило ефективну обробку запитів та інтеграцію з необхідними API для зберігання та отримання даних опитувань. Для зберігання і управління даними була використана база даних MySQL, що забезпечила надійність та швидкодію операцій з інформацією.

Проєкт пройшов інтенсивне тестування з використанням автоматизованих засобів, що дозволило підтвердити коректність роботи всіх функціональних можливостей, а також стабільність вебсервісу при різних умовах використання.

У майбутньому рекомендується продовжувати розвивати вебсервіс шляхом розширення функціоналу, вдосконалення інтерфейсу користувача та оптимізації продуктивності. Важливо також здійснювати систематичний моніторинг та підтримку для забезпечення безперебійної роботи сервісу та задоволення потреб користувачів. Проєкт дозволяє класичним користувачам проводити опитування та тести у новому форматі, забезпечуючи сучасним функціональні інструменти для зручного та ефективного взаємодії з даними.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційна документація React. URL: https://legacy.reactjs.org/docs/getting-started.html (дата звернення: 09.03.2024).
2. Офіційна документація Redux. URL: https://redux.js.org (дата звернення: 19.03.2024).
3. Офіційна документація Firebase. URL: https://dev.mysql.com/doc/ (дата звернення: 21.03.2024).
4. GitHub репозиторії з прикладами використання Firebase з React. URL: https://firebase.google.com (дата звернення: 05.04.2024).
5. Статті на Medium та блоги React і Firebase розробників. URL: https://blogiantic.medium.com/creating-a-website-using-reactjs-and-firebase-abc20be816bc (дата звернення: 11.04.2024).
6. Курси з React та Firebase на платформах навчання. URL: http://surl.li/urwcf (дата звернення: 27.04.2024).
7. Офіційна документація та приклади використання Jest для тестування React. URL: http://surl.li/urwch (дата звернення: 05.05.2024).
8. Курси з тестування React на платформах навчання. URL: http://surl.li/urwci (дата звернення: 12.05.2024).
9. Офіційна документація з Cypress для тестування функціональності вебсервісу. URL: https://docs.cypress.io/guides/overview/why-cypress (дата звернення: 12.05.2024).
10. Статті та ресурси з відгуками та рекомендаціями щодо підходів до розробки вебсервісів з React, Redux і Firebase. URL: https://dev.to/mliakos/real-time-app-using-react-redux-tailwind-css-firebase-part-2-1cgk (дата звернення: 13.05.2024).