МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ПРОГРАМНИЙ КОМПЛЕКС АНКЕТУВАННЯ З МОЖЛИВІСТЮ ФУНКЦІЮВАННЯ КЛІЄНТА В ОФЛАЙН РЕЖИМІ В УМОВАХ ОБМЕЖЕНОГО ЧИ ВІДСУТНЬОГО ЗВ'ЯЗКУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010225

Виконав студент 4-го курсу, групи 402
_____ А. О. Топчій
«17» червня 2024 р.

Керівник: канд. техн. наук, доцент
_____Є. В. Сіденко
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Топчію Артему Олександровичу.

1. Тема кваліфікаційної роботи «Програмний комплекс анкетування з можливістю функціювання клієнта в офлайн режимі в умовах обмеженого чи відсутнього зв'язку».

Керівник роботи Савінов Володимир Юрійович, доцент кафедри КІ, канд. техн. наук.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: оцінки наявних на ринку інструментів для створення та проведення анкетувань; технічні вимоги до забезпечення безпеки та конфіденційності даних; вимоги до роботи в умовах обмеженого або відсутнього інтернет-з'єднання; думки експертів щодо необхідного функціоналу у застосунку.

Очікуваний результат: програмний комплекс анкетування з можливістю функціювання клієнта в офлайн режимі в умовах обмеженого чи відсутнього зв'язку.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз сучасного стану задачі створення анкетувальних систем з можливістю функціонування в офлайн режимі;

– огляд наявних на ринку інструментів для створення та проведення анкетувань;

– вибір технологій та інструментів для розробки вебзастосунку, включаючи фронтенд, бекенд та бази даних;

– проектування архітектури вебзастосунку, що забезпечує роботу в офлайн режимі та синхронізацію даних при підключенні до інтернету;

– тестування та перевірка працездатності системи в умовах обмеженого інтернет-з'єднання.

5. Перелік графічного матеріалу: 40 ілюстрацій, 1 таблиця, презентація.

6. Завдання до спеціальної частини: «Охорона праці та забезпечення безпеки при розробці та використанні програмного комплексу анкетування».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А.О. , доцент кафедри екології	

Керівник роботи __ доцент кафедри КІ, канд. техн. наук Савінов Володимир Юрійович _____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Топчій А. О._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Програмний комплекс анкетування з можливістю функціювання клієнта в офлайн режимі в умовах обмеженого чи відсутнього зв'язку

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: Аналіз сучасного стану задачі створення анкетувальних комплексів, огляд існуючих інструментів, розробка ПЗ	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	

Розробив студент __Топчій А. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ доцент кафедри КІ, канд. техн. наук Савінов Володимир Юрійович
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи бакалавра групи 402 ЧНУ ім. Петра Могили
Топчія Артема Олександровича
Тема: «Програмний комплекс анкетування з можливістю функціювання клієнта в офлайн режимі в умовах обмеженого чи відсутнього зв'язку»

Актуальність роботи полягає в необхідності забезпечення ефективного збору даних у регіонах з обмеженим або відсутнім доступом до інтернету. Це особливо важливо в контексті відновлення інфраструктури після військових дій та стихійних лих, коли оперативний збір даних про потреби населення є критичним для планування та надання допомоги.

Об'єкт роботи – процес збору та обробки даних про потреби населення в умовах обмеженого зв'язку.

Предмет роботи – програмні рішення для забезпечення функціювання анкетування в офлайн режимі та синхронізації даних з сервером після відновлення зв'язку.

Метою кваліфікаційної роботи є розробка програмного комплексу для анкетування, який здатен функціонувати в офлайн режимі, з можливістю автоматичної синхронізації даних з сервером після відновлення інтернет-з'єднання, та забезпечення можливості спільного редагування анкет різними користувачами.

Пояснювальна записка складається зі вступу, трьох розділів, висновків та додатків.

У першому розділі розглядається актуальність задачі, огляд та аналіз наявних аналогів та публікацій з теми анкетування та збору даних у складних умовах, а також вимоги до програмного забезпечення.

У другому розділі описані методи та технології для розробки системи анкетування, зокрема використання PouchDB та CouchDB для офлайн зберігання та синхронізації даних, а також технології React, Redux, та Node.js для створення клієнтської та серверної частин програмного комплексу.

У третьому розділі детально описується процес розробки програмного комплексу, включаючи проектування архітектури системи, реалізацію функції офлайн зберігання даних та синхронізації, можливості спільного редагування анкет, а в кінці розділу наведено результати тестування та апробації системи, описані результати її роботи в реальних умовах.

В результаті роботи було розроблено програмний комплекс анкетування, який забезпечує ефективний збір та обробку даних в умовах обмеженого або відсутнього інтернет-з'єднання, що дозволяє оперативно реагувати на потреби населення в кризових ситуаціях.

Кваліфікаційна робота містить 66 сторінок, 40 рисунків, 1 таблицю, 25 використаних джерел та 3 додатка.

Ключові слова: офлайн анкетування, збір даних, синхронізація даних, React, PouchDB, CouchDB, Node.js.

ABSTRACT

Bachelor's qualification work of a student of group 402 of ChNU named after Petro Mohyla Black Sea National University

Topchii Artem

Topic: "Survey software complex with client functionality in offline mode under conditions of limited or no connectivity"

The relevance of this work lies in the necessity of ensuring effective data collection in regions with limited or no internet access. This is especially important in the context of infrastructure recovery after military actions and natural disasters, where rapid data collection on the needs of the population is critical for planning and providing aid.

The object of the study is the process of collecting and processing data on the needs of the population under conditions of limited connectivity.

The subject of the research is software solutions that ensuring the functionality of surveying in offline mode and data synchronization with the server after connection restoration.

The purpose of the qualification work is to develop a software complex for surveying that can operate in offline mode, with the capability of automatic data synchronization with the server after internet connection restoration, and to ensure the possibility of collaborative editing of surveys by different users.

The explanatory note consists of an introduction, three chapters, conclusions, and appendices.

In the first chapter, the relevance of the task, review, and analysis of existing analogs and publications on the topic of surveying and data collection in difficult conditions, as well as software requirements, are considered.

The second chapter describes the methods and technologies for developing the survey system, particularly the use of PouchDB and CouchDB for offline data storage and synchronization, as well as React, Redux, and Node.js technologies for creating the client and server parts of the software complex.

The third chapter details the process of developing the software complex, including system architecture design, implementation of offline data storage and synchronization functions, collaborative survey editing capabilities, and presents the results of system testing and trials, describing its performance in real conditions.

As a result of the work, a survey software complex was developed, which ensures effective data collection and processing under conditions of limited or no internet connectivity, allowing for prompt response to the needs of the population in crisis situations.

The qualification work contains 66 pages, 40 figures, 1 table, 25 used sources and 3 appendices.

Keywords: offline surveying, data collection, data synchronization, React, PouchDB, CouchDB, Node.js.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 6
ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ВИЗНАЧЕННЯ АКТУАЛЬНОСТІ РОЗРОБКИ СИСТЕМИ ДЛЯ АНКЕТУВАННЯ ТА ПОСТАНОВКА ЗАДАЧІ ... 10
1.1 Опис предметної області та актуальність 10
1.2 Огляд та аналіз наявних аналогів та публікацій 11
1.3 Вимоги до програмного забезпечення, постановка задачі 19
Висновки до розділу 1 21
2 МЕТОДИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ СИСТЕМИ АНКЕТУВАННЯ 23
2.1 Технології для розробки вебзастосунку 23
2.2 Методи для забезпечення офлайн-функціоналу 25
Висновки до розділу 2 26
3 СТВОРНІЯ МАКЕТУ, ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ АНКЕТУВАННЯ ТА ТЕСТУВАННЯ 27
3.1 Розробка дизайну вебзастосунку у редакторі Figma 27
3.2 Опис програмної реалізації 30
3.2 Тестування веб-застосунку 51
Висновки до розділу 3 57
ВИСНОВКИ 59
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 61
ДОДАТОК А СТОРІНКА HOME 63
ДОДАТОК Б ФУНКЦІЯ ОНОВЛЕННЯ СПИСКУ ОПИТУВАНЬ 65
ДОДАТОК В ФУНКЦІЯ ВИДАЛЕННЯ ВСІХ ОПИТУВАНЬ 66

ПЕРЕЛІК СКОРОЧЕНЬ

ШІ - Штучний Інтелект

JWT - JSON Web Tokens

NPM – Node Package Manager

NPX - Node Package eXecute

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

CSS - Cascading Style Sheets

API - Application Programming Interface

SVG - Scalable Vector Graphics

DOM - Document Object Model

CORS - Cross-Origin Resource Sharing

Fs - File System(модуль Node.js)

ВСТУП

З початку російсько-української війни Україна стикається з масштабними руйнуваннями житлового фонду, що потребує значної фінансової підтримки для відновлення. Одним із важливих аспектів залучення та ефективного розподілу цієї допомоги є точне визначення потреб постраждалого населення. Це особливо актуально для сільських місцевостей, де мешкають переважно люди похилого віку, які часто не мають навичок користування сучасними технологіями.

Для точного оцінювання збитків та визначення пріоритетів у наданні допомоги, міжнародні донори часто вимагають детальні відповіді від мешканців постраждалих регіонів. Анкетування дозволяє зібрати необхідні дані про ступінь руйнувань житлових будинків, потреби в ремонті, а також інші важливі аспекти життєдіяльності, такі як доступ до медичних послуг, водопостачання та електроенергії.

Основним викликом у проведенні анкетування в сільських регіонах є те, що більшість мешканців цих районів — люди похилого віку, які не завжди володіють комп’ютерними навичками. Вони часто не мають доступу до інтернету або сучасних пристроїв, що ускладнює процес збору даних. В таких умовах важливо організувати виїзні бригади спеціалістів гуманітарних організацій, які можуть провести анкетування безпосередньо на місцях.

Виїзне анкетування дозволяє зібрати точні та актуальні дані безпосередньо від постраждалих людей. Спеціалісти через спілкування з людьми можуть дізнаватися необхідну інформацію про руйнування, різноманітні пошкодження, дані людей та заносити її до бази даних, заповнюючи форму. Це дає змогу донести до донорів реальну картину і визначити конкретні потреби для кожного населеного пункту.

Комплексне анкетування є важливим інструментом для координації допомоги та оптимізації ресурсів. Воно допомагає:

– визначити пріоритети відновлення. На основі зібраних даних можна визначити найбільш постраждалі регіони та розподілити ресурси ефективніше;

– забезпечити прозорість процесу. Чітка і структурована інформація допомагає донорським організаціям відслідковувати використання коштів і запобігати можливим зловживанням;

– підвищити ефективність допомоги. Спеціалісти можуть надавати цільову допомогу саме там, де вона найбільше потрібна, враховуючи специфічні потреби кожного домогосподарства.

В умовах війни та постійних руйнувань, тема комплексного анкетування є надзвичайно актуальною. Вона дозволяє забезпечити оперативний збір даних, необхідних для планування та реалізації програм відновлення. Виїзні бригади спеціалістів, які проводять анкетування на місцях, є важливим елементом цього процесу, забезпечуючи доступ до інформації навіть у найвіддаленіших куточках країни.

Таким чином, впровадження системи комплексного анкетування не лише допомагає краще розподілити ресурси, але й підвищує рівень довіри донорів до процесу відновлення, сприяючи більш ефективному відновленню України.

Об’єктом дослідження є процеси збору та обробки даних в умовах обмеженого чи відсутнього інтернет-з'єднання.

Предметом дослідження є технології та методи розробки програмного комплексу для анкетування, який може функціонувати в офлайн режимі.

Метою роботи є розробка програмного комплексу для анкетування з можливістю функціювання в офлайн режимі, що дозволить збирати та обробляти дані в умовах обмеженого чи відсутнього зв'язку.

Для досягнення цієї мети необхідно:

1) проаналізувати сучасний стан задачі збору даних для анкетування в умовах обмеженого інтернет-з'єднання;

2) дослідити існуючі системи та публікації, пов'язані з анкетуванням та збором даних;

3) визначити вимоги до програмного забезпечення для забезпечення ефективного функціонування в офлайн режимі;

4) розробити архітектуру та структуру програмного комплексу, який забезпечить збір та обробку даних в умовах обмеженого зв'язку;

5) реалізувати програмний комплекс та провести його тестування для забезпечення надійної роботи в офлайн режимі;

6) провести порівняльний аналіз ефективності розробленого комплексу з існуючими рішеннями на ринку.

У першому розділі розглядається сучасний стан задачі збору даних для анкетування, огляд наявних рішень даної проблеми та публікацій, а також визначаються задачі, які необхідно виконати.

Другий розділ містить у собі огляд існуючих технологій та методів, а також аналіз моделей для поставленої задачі.

У третьому розділі описується підхід до проектування архітектури інформаційної системи, її реалізація та тестування, а також порівняння з існуючими рішеннями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ВИЗНАЧЕННЯ АКТУАЛЬНОСТІ РОЗРОБКИ СИСТЕМИ ДЛЯ АНКЕТУВАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області та актуальність

Анкетування є важливим інструментом для збору даних про ступінь руйнувань, потреби в ремонті та інші аспекти життєдіяльності, такі як доступ до медичних послуг, водопостачання та електроенергії [1]. Міжнародні донори часто вимагають детальні відповіді від мешканців постраждалих регіонів для точного оцінювання збитків та визначення пріоритетів у наданні допомоги. Однак, у сільських регіонах, де більшість мешканців не володіють комп’ютерними навичками і не мають доступу до інтернету, цей процес стає надзвичайно складним.

Основним викликом у проведенні анкетування в таких умовах є організація виїзних бригад спеціалістів гуманітарних організацій, які можуть провести анкетування безпосередньо на місцях. Виїзне анкетування дозволяє зібрати точні та актуальні дані безпосередньо від постраждалих людей. Спеціалісти через спілкування з людьми можуть дізнаватися необхідну інформацію про руйнування, різноманітні пошкодження та потреби людей, заносячи її до бази даних, заповнюючи форму. Це дає змогу донести до донорів реальну картину і визначити конкретні потреби для кожного населеного пункту.

Комплексне анкетування є важливим інструментом для координації допомоги та оптимізації ресурсів. Воно допомагає визначити пріоритети відновлення, забезпечити прозорість процесу та підвищити ефективність допомоги. На основі зібраних даних можна визначити найбільш постраждалі регіони та розподілити ресурси ефективніше. Чітка і структурована інформація допомагає донорським організаціям відслідковувати використання коштів і запобігати можливим зловживанням. Спеціалісти можуть надавати цільову допомогу саме там, де вона найбільше потрібна, враховуючи специфічні потреби кожного домогосподарства.

Завдяки впровадженню системи комплексного анкетування можна забезпечити оперативний збір даних, необхідних для планування та реалізації програм відновлення. Виїзні бригади спеціалістів, які проводять анкетування на місцях, є важливим елементом цього процесу, забезпечуючи доступ до інформації навіть у найвіддаленіших куточках країни.

В умовах війни та постійних руйнувань, тема комплексного анкетування є надзвичайно актуальною. Вона дозволяє забезпечити оперативний збір даних, необхідних для планування та реалізації програм відновлення. Виїзні бригади спеціалістів, які проводять анкетування на місцях, є важливим елементом цього процесу, забезпечуючи доступ до інформації навіть у найвіддаленіших куточках країни.

Таким чином, впровадження системи комплексного анкетування не лише допомагає краще розподілити ресурси, але й підвищує рівень довіри донорів до процесу відновлення, сприяючи більш ефективному відновленню України. У цьому контексті особливої актуальності набуває розробка програмного комплексу для анкетування з можливістю функціювання в офлайн режимі. Це дозволяє забезпечити безперебійну роботу системи в умовах обмеженого або відсутнього інтернет-з'єднання.

Предметом дослідження є технології та методи розробки програмного комплексу для анкетування, який функціонує в офлайн режимі.

Розробка програмного комплексу для анкетування з можливістю функціювання в офлайн режимі є актуальною та важливою задачею, яка дозволить ефективно збирати та обробляти дані в умовах обмеженого чи відсутнього зв'язку, що сприятиме більш ефективному відновленню України та забезпеченню необхідної підтримки постраждалим.

1.2 Огляд та аналіз наявних аналогів та публікацій

Для здійснення анкетування існує багато різних рішень, які можна умовно поділити на дві категорії: стабільні та популярні засоби анкетування, а також

вузькоспеціалізовані та малопопулярні. Однак більшість рішень з обох категорій не повністю відповідають умовам України через брак тих чи інших функцій.

1. Стабільні та популярні засоби анкетування. Вони широко використовуються завдяки своїй надійності та доступності. Приклади таких платформ: Google Forms, Microsoft Forms тощо.

2. Вузькоспеціалізовані та малопопулярні засоби анкетування. Вони призначені для більш специфічних завдань і часто використовуються у гуманітарних місіях та дослідницьких проектах. Приклади таких платформ: KoboToolbox, LimeSurvey, OhMyForm тощо.

Кожен з цих типів рішень має свої переваги та недоліки, які розглянуті далі.

Стабільні та популярні засоби анкетування мають такі **переваги**:

– інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати анкети без спеціальних знань;

– легко інтегруються з іншими популярними сервісами, такими як Google Drive або Microsoft Office 365, що спрощує обробку даних;

– ці платформи безкоштовні або мають доступні тарифи, що робить їх привабливими для широкого кола користувачів.

Стабільні та популярні засоби анкетування мають такі **недоліки**:

– відсутність деяких просунутих функцій, таких як геоприв'язка або офлайн-режим, що може бути критично важливим для анкетування в сільській місцевості;

– вимагають стабільного інтернет-з'єднання, що не завжди є можливим у віддалених районах України;

– обмежений набір опцій для персоналізації та адаптації форм під специфічні потреби гуманітарних організацій.

Вузькоспеціалізовані та малопопулярні засоби анкетування мають такі **переваги**:

– підтримують більш складні типи питань, логіку гілок, офлайн-режим та інші функції, що корисні для збору даних у польових умовах;

– можливість детально налаштовувати форми та опитування під специфічні потреби користувачів;

– здатність працювати без інтернету, що дозволяє проводити анкетування в регіонах з нестабільним або відсутнім інтернет-з'єднанням.

Вузькоспеціалізовані та малопопулярні засоби анкетування мають такі **недоліки**:

– інтерфейс може бути менш інтуїтивним, вимагаючи від користувачів більше часу на навчання;

– менша поширеність означає менше ресурсів для підтримки та менше готових рішень;

– вузькоспеціалізовані платформи можуть мати обмежені можливості інтеграції з іншими сервісами та інструментами.

Таблиця 2.1 – Порівняння платформ для проведення анкетування

Категорія	**Переваги**	**Недоліки**
Стабільні та популярні засоби анкетування (Google Forms, Microsoft Forms)	– інтуїтивний інтерфейса; – сумісні з популярними сервісами (Google Drive, Office 365); – безкоштовні або недорогі.	– відсутність геоприв'язки, офлайн-режиму; – потрібне стабільне з'єднання; – обмежені опції персоналізації.
Вузькоспеціалізовані та малопопулярні засоби анкетування (KoboToolbox, LimeSurvey, OhMyForm)	– підтримка складних питань, офлайн-режиму; – детальна адаптація форм; – можливість працювати без інтернету.	– потрібен час на освоєння; – менше підтримки та готових рішень; – менша сумісність з іншими сервісами.

Одним з лідерів ринку для використання гуманітарними організаціями є KoboToolbox (рис. 1.1). Даний ресурс дозволяє проводити анкетування і збір даних у складних умовах, що робить його гарним інструментом для роботи в польових умовах. Основні переваги KoboToolbox включають підтримку офлайн-режиму, що є критично важливим для роботи у віддалених або постраждалих районах. Дані можуть бути зібрані на мобільних пристроях і синхронізовані з сервером пізніше, коли з'явиться інтернет-з'єднання. Платформа підтримує складні типи питань, логіку гілок, мультимедійні додатки (фото, відео, аудіо), а також геоприв'язку, що дозволяє отримувати більш детальну і якісну інформацію [2].

Рисунок 1.1 – Логотип компанії KoboToolbox

Висока гнучкість налаштувань дозволяє користувачам детально налаштовувати форми під специфічні потреби, включаючи створення кастомізованих питань і налаштування логіки опитувань. KoboToolbox забезпечує високу безпеку зібраних даних, що особливо важливо для захисту особистої інформації респондентів у гуманітарних контекстах. Платформа має відкритий код, що дозволяє користувачам адаптувати її під свої потреби і інтегрувати з іншими системами. Проте через розширені функції і можливості платформи новим користувачам може знадобитися час на навчання і освоєння, оскільки інтерфейс не настільки інтуїтивний, як у деяких популярних платформ. У порівнянні з більш популярними рішеннями, такими як Google Forms (рис. 1.2), підтримка і ресурси для навчання можуть бути обмежені. Також до недоліків цієї платформі можна віднести одну важливу функцію – одночасне редагування (створення) анкети в польових умовах з двох різних пристроїв двома різними спеціалістами. Така необхідність часто виникає, коли час обмежений і в той час як один зі спеціалістів здійснює збір та внесення у форму текстових даних, яку він отримує спілкуючись

з особою, інший спеціаліст міг би, наприклад, фотографувати пошкодження житлового приміщення і відразу ж прикріплювати відповідні медіаматеріали до анкети.

Рисунок 1.2 – Логотип Google Forms

Одним з популярних інструментів для проведення анкетування є Google Forms [3]. Цей ресурс відомий своєю легкістю використання та інтеграцією з іншими сервісами Google, що робить його зручним для швидкого створення та поширення анкет. Основні переваги Google Forms включають інтуїтивно зрозумілий інтерфейс, який дозволяє навіть недосвідченим користувачам швидко створювати анкети та форми без необхідності спеціальних знань. Інтеграція з іншими сервісами Google, такими як Google Drive та Google Sheets, дозволяє легко зберігати та аналізувати зібрані дані. Платформа є безкоштовною, що робить її доступною для широкого кола користувачів, включаючи освітні та некомерційні організації.

Однак, Google Forms має свої обмеження. Відсутність підтримки офлайн-режиму означає, що для використання платформи необхідне стабільне інтернет-з'єднання, що може бути проблематичним у віддалених або постраждалих районах. Крім того, платформа має обмежені можливості налаштування та персоналізації форм. Користувачі не можуть використовувати складні типи питань або логіку гілок, що обмежує можливості для збору детальної інформації. Відсутність геоприв'язки та мультимедійних додатків також знижує ефективність використання Google Forms у польових умовах.

Також Google Forms не підтримує функцію одночасного редагування форми з різних пристроїв різними користувачами у реальному часі, що може бути критичним для деяких сценаріїв використання, таких як робота гуманітарних організацій на місцях. Наприклад, коли один спеціаліст збирає текстові дані, спілкуючись з особою, інший спеціаліст міг би одночасно фотографувати пошкодження та прикріплювати медіаматеріали до тієї ж анкети.

Одним з інструментів для проведення анкетування є Microsoft Forms (рис. 1.3). Цей сервіс інтегрований з іншими продуктами Microsoft, такими як Office 365, що робить його зручним для користувачів, які вже працюють у цій екосистемі [4].

Рисунок 1.3 – Логотип Microsoft Forms

Основні переваги Microsoft Forms включають інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати анкети та форми без потреби в спеціальних навичках. Інтеграція з OneDrive та Excel спрощує зберігання і аналіз зібраних даних, а також надає можливість автоматизації процесів за допомогою Power Automate.

Однак Microsoft Forms має свої обмеження. Відсутність офлайн-режиму обмежує його використання в умовах, де немає стабільного інтернет-з'єднання. Платформа також має обмежені можливості налаштування форм та підтримує менш складні типи питань порівняно з вузькоспеціалізованими інструментами, такими як KoboToolbox. Відсутність геоприв'язки та мультимедійних додатків знижує ефективність використання Microsoft Forms для детального збору даних у польових умовах. Крім того, платформа не підтримує одночасне редагування форм з різних пристроїв у реальному часі, що може бути критичним для спільної роботи

спеціалістів у польових умовах.

Ще одним із популярних інструментів для проведення анкетування є LimeSurvey (рис. 1.4). Цей ресурс відомий своєю гнучкістю та потужними можливостями для створення складних опитувань

Рисунок 1.4 – Логотип LimeSurvey

Основні переваги LimeSurvey включають високу ступінь налаштування анкет, що дозволяє користувачам створювати опитування з багаторівневою логікою гілок, різноманітними типами питань та умовами відображення. Це робить LimeSurvey ідеальним інструментом для проведення детальних досліджень та збору складної інформації.

Однією з ключових переваг LimeSurvey є можливість розгортання на власних серверах, що забезпечує повний контроль над даними та їх безпекою. Це особливо важливо для організацій, які працюють з конфіденційною інформацією або мають специфічні вимоги щодо зберігання даних. Крім того, LimeSurvey підтримує мультимовні опитування, що дозволяє легко охоплювати міжнародну аудиторію.
Однак, LimeSurvey має і деякі обмеження. Платформа для багатьох, в тому числі і спеціалістів, які будуть створювати та налаштовувати форму, може здатися складною через велику кількість налаштувань та можливостей. Вона вимагає певного рівня технічної підготовки для ефективного використання всіх її функцій. Також LimeSurvey не є безкоштовним у повному обсязі: хоча існує безкоштовна версія, для доступу до всіх функцій необхідно придбати платний тарифний план.
Ще одним обмеженням є потреба в адмініструванні сервера у випадку самостійного розгортання LimeSurvey, що може вимагати додаткових ресурсів та технічних знань. Тим не менш, для багатьох організацій ці витрати виправдані завдяки широким можливостям платформи для створення індивідуалізованих опитувань та

глибокого аналізу зібраних даних.

Одним із популярних інструментів для проведення анкетування є OhMyForm (рис. 1.4). Цей ресурс відомий своєю відкритістю та можливістю повної налаштування під потреби користувачів.

Рисунок 1.5 – Логотип OhMyForm

Основні переваги OhMyForm включають доступ до вихідного коду, що дозволяє користувачам адаптувати платформу під специфічні вимоги своїх проектів. Це робить OhMyForm ідеальним вибором для організацій, які потребують гнучкості та контролю над своїми опитуваннями.

Однією з ключових переваг OhMyForm є можливість розгортання на власних серверах, що забезпечує повний контроль над даними та їх безпекою. Це особливо важливо для організацій, які працюють з конфіденційною інформацією або мають специфічні вимоги щодо зберігання даних. Платформа також підтримує мультимовні опитування, що дозволяє легко охоплювати міжнародну аудиторію.
Однак, OhMyForm має і деякі обмеження. Для новачків платформа може здатися складною через необхідність адміністрування сервера та налаштування системи. Вона вимагає певного рівня технічної підготовки для ефективного використання всіх її функцій. Також, хоча платформа є безкоштовною та з відкритим кодом, для доступу до деяких розширених функцій або підтримки може знадобитися залучення додаткових ресурсів.

Ще одним обмеженням є обмежена кількість готових шаблонів та інтеграцій у порівнянні з деякими комерційними платформами, що може вимагати більше часу на створення опитувань з нуля. Тим не менш, для багатьох організацій ці витрати виправдані завдяки можливості повної кастомізації та розширенню

функціональності OhMyForm відповідно до потреб конкретного проекту.

За останні роки з’являється все більше подібних сервісів для створення форм та проведення анкетування. Вони набувають все ширшого і ширшого функціоналу, проте все одно не охоплюють усіх можливих сценаріїв використання та особливо не пристосовані для роботи в польових умовах України.

Отже, існуючі рішення мають ряд суттєвих недоліків у функціоналі: складність використання при створенні форми (KoboToolbox), відсутність можливості працювати в умовах поганого або відсутнього зв’язку(Google Forms, Microsoft Forms), обмежена кількість готових шаблонів (OhMyForms). Також, жодне з розглянутих рішень взагалі не має функціоналу, який дозволив би працювати з однією анкетою одночасно двом співробітникам. Тому розробка нового спеціалізованого вебзастосунку з урахуванням недоліків вже існуючих платформ є цілком виправданою і дозволить суттєво розширити функціонал та головне – скоротити час на збір даних, а, отже, пришвидшити надання цих даних партнерам і швидше отримувати від них допомогу.

1.3 Вимоги до програмного забезпечення, постановка задачі

Для успішного впровадження та функціонування системи анкетування з можливістю роботи в офлайн режимі, необхідно розробити програмне забезпечення, що задовольняє ряд специфічних вимог. Це дозволить забезпечити стабільну роботу системи в умовах обмеженого або відсутнього інтернет-з'єднання, а також ефективне взаємодію користувачів.

Вимоги до програмного забезпечення:

1) офлайн зберігання даних;
2) зручність використання;
3) безпека та конфіденційність даних;
4) ефективність та надійність.

Програмне забезпечення повинно дозволяти користувачам зберігати введені дані анкет на локальному пристрої. Це є критично важливим у випадках, коли

інтернет-з'єднання відсутнє або нестабільне. Використання технологій, таких як PouchDB та CouchDB, забезпечить надійне зберігання даних до моменту відновлення зв'язку [5]. Після відновлення інтернет-з'єднання, програмне забезпечення повинно автоматично синхронізувати локально збережені дані з сервером. Це забезпечить актуальність даних на сервері та мінімізує ризик втрати інформації.

Програмне забезпечення повинно підтримувати функцію спільного редагування, дозволяючи двом або більше користувачам одночасно вносити зміни до однієї анкети. Це може бути реалізовано за допомогою WebSockets, що забезпечує реального часу синхронізацію змін між пристроями [6]. У випадку одночасного редагування однієї анкети кількома користувачами, система повинна мати механізми вирішення конфліктів, щоб запобігти втраті або перекриттю даних. Це може включати версіонування даних або механізми блокування [7].

Інтерфейс користувача повинен бути максимально простим та зрозумілим, особливо враховуючи, що більшість користувачів – люди похилого віку, які можуть не мати достатніх навичок роботи з сучасними технологіями [8]. Використання відомих шаблонів UX/UI, таких як Material Design, сприятиме легкості використання [9]. Програмне забезпечення повинно коректно працювати на різних пристроях, включаючи смартфони, планшети та настільні комп'ютери. Це забезпечить зручність використання незалежно від типу пристрою.

Всі дані повинні бути захищені від несанкціонованого доступу. Використання шифрування даних як на локальних пристроях, так і під час передачі між клієнтом і сервером є обов'язковим [10,11]. Для доступу до системи необхідно забезпечити надійні механізми аутентифікації та авторизації користувачів. Використання JWT (JSON Web Tokens) дозволить безпечно керувати сесіями користувачів [12].

Програмне забезпечення повинно швидко реагувати на дії користувача і забезпечувати високу швидкість обробки даних, навіть в умовах великого

навантаження. Система повинна бути стійкою до збоїв та помилок, з можливістю автоматичного відновлення після них.

На основі визначених вимог, основними задачами розробки програмного забезпечення є:

1) аналіз вимог користувачів та специфікацій системи. Визначення функціональних та нефункціональних вимог до програмного забезпечення на основі аналізу існуючих рішень та потреб користувачів;

2) розробка архітектури програмного забезпечення. Проектування системи, яка включає компоненти для зберігання даних в офлайн режимі, синхронізації даних, спільного редагування та управління конфліктами;

3) реалізація клієнтської та серверної частини [13,14]. Розробка інтерфейсу користувача з використанням технологій React, Redux, та Service Workers для забезпечення офлайн режиму [15]. Реалізація серверної частини з використанням Node.js, Express та PouchDB з CouchDB [16,17];

4) інтеграція механізмів безпеки. Впровадження шифрування даних, аутентифікації та авторизації користувачів;

5) тестування та оптимізація. Проведення функціонального та навантажувального тестування системи, а також оптимізація її продуктивності;

Таким чином, виконання цих задач дозволить створити надійний і ефективний програмний комплекс для анкетування, який забезпечить збір та обробку даних в умовах обмеженого або відсутнього інтернет-з'єднання, а також можливість спільного редагування анкет різними користувачами.

Висновки до розділу 1

В рамках першого розділу було проаналізовано предметну область анкетування, визначено його актуальність та важливість для відновлення житлового фонду в Україні після руйнувань, спричинених війною. Анкетування є ключовим інструментом для збору даних про ступінь пошкоджень, потреби в ремонті та інші життєво важливі аспекти, такі як доступ до медичних послуг,

водопостачання та електроенергії. Особливо актуальним цей процес є для сільських регіонів, де більшість населення становлять люди похилого віку, які часто не мають навичок роботи з сучасними технологіями. Основною проблемою в проведенні анкетування в таких умовах є необхідність організації виїзних бригад спеціалістів, які можуть проводити анкетування безпосередньо на місцях. Виїзне анкетування дозволяє зібрати точні та актуальні дані безпосередньо від постраждалих людей, що дає змогу донести до донорів реальну картину і визначити конкретні потреби для кожного населеного пункту. Це забезпечує координацію допомоги та оптимізацію ресурсів, визначаючи пріоритети відновлення і забезпечуючи прозорість процесу. У процесі дослідження були розглянуті існуючі рішення для анкетування, які можна поділити на стабільні та популярні засоби, а також вузькоспеціалізовані та малопопулярні інструменти. Виявлено, що жодне з цих рішень не повністю відповідає потребам в умовах України через брак необхідних функцій, таких як офлайн-режим або можливість спільного редагування анкет.

На основі проведеного аналізу було визначено, що розробка спеціалізованого вебзастосунку з урахуванням недоліків існуючих платформ є виправданою і необхідною. Такий застосунок дозволить ефективно збирати та обробляти дані навіть в умовах обмеженого або відсутнього інтернет-з'єднання, а також забезпечить можливість спільного редагування анкет різними користувачами.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ СИСТЕМИ АНКЕТУВАННЯ

2.1 Технології для розробки вебзастосунку

Розробка вебзастосунку для анкетування з можливістю функціювання в офлайн режимі вимагає використання сучасних технологій, які забезпечать необхідну функціональність, надійність та ефективність системи. Нижче наведено основні технології, які планується використовувати для створення цього програмного комплексу.

1) **React**. React – це популярна бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена компанією Facebook. Вона дозволяє розробникам створювати швидкі та інтерактивні вебзастосунки за допомогою компонентного підходу. Основні переваги React:

– компонентний підхід. Інтерфейс користувача розбивається на окремі компоненти, кожен з яких має свою логіку та стан. Це сприяє повторному використанню коду та спрощує його підтримку;

– віртуальний DOM [18]. React використовує віртуальний DOM для мінімізації кількості операцій з реальним DOM, що забезпечує високу продуктивність та швидкість реакції інтерфейсу на дії користувача;

– управління станом. Використання бібліотек для управління станом, таких як Redux або Context API, дозволяє ефективно керувати даними в застосунку, що є важливим для спільного редагування анкет;

2) **Redux**. Redux – це бібліотека для управління станом у JavaScript-застосунках, яка зазвичай використовується разом з React. Основні переваги Redux:

– централізоване управління станом. Всі дані застосунку зберігаються у єдиному сховищі, що спрощує відстеження змін та управління станом;

– передбачуваність. Завдяки чітко визначеним правилам зміни стану (редюсери), поведінка застосунку стає передбачуваною, що спрощує відлагодження та тестування;

– інструменти розробки. Redux DevTools надає потужні інструменти для відстеження змін стану та відлагодження застосунку;

3) **React Query**. React Query – це бібліотека для управління даними сервера у React-застосунках. Вона надає потужні інструменти для фетчінгу, кешування, синхронізації та оновлення даних у режимі реального часу [19]. Основні переваги React Query:

– автоматичне оновлення даних. Дані автоматично оновлюються при зміні стану мережі або отриманні нових даних з сервера;

– кешування. Дані кешуються локально, що дозволяє зменшити кількість запитів до сервера та підвищити продуктивність;

– управління станом. React Query спрощує управління станом даних, забезпечуючи синхронізацію даних між різними компонентами застосунку;

4) **Node.js з Express**. Node.js – це середовище виконання JavaScript, яке дозволяє розробляти серверні додатки. Express – це мінімалістичний веб-фреймворк для Node.js, який спрощує розробку серверної частини. Основні переваги Node.js та Express:

– одночасна обробка запитів. Node.js використовує подієву модель, що дозволяє обробляти багато запитів одночасно без блокування;

– швидкодія. Висока продуктивність завдяки використанню движка V8;

– широка екосистема. Наявність великої кількості модулів та пакетів у npm (Node Package Manager);

5) **JWT (JSON Web Tokens)**. JWT – це стандарт для створення токенів доступу, які використовуються для аутентифікації та авторизації користувачів. Основні переваги JWT:

– безпека. Токени підписуються криптографічними методами, що забезпечує захист від підробки;

– легкість використання. Токени можуть бути збережені на клієнтському пристрої і передані з кожним запитом до сервера;

– масштабованість. JWT легко інтегруються з різними серверами та клієнтами, що забезпечує гнучкість системи.

2.2 Методи для забезпечення офлайн-функціоналу

Забезпечення офлайн-функціоналу є ключовим елементом для вебзастосунку анкетування, особливо в умовах обмеженого або відсутнього інтернет-з'єднання. Офлайн-функціонал дозволяє зберігати дані локально на пристрої користувача та синхронізувати їх з сервером, коли з'явиться інтернет-з'єднання. Для вебзастосунку анкетування, зважаючи на вимоги до офлайн-роботи та простоти реалізації, найбільш підходящим варіантом буде використання **PouchDB у поєднанні з CouchDB**. Це рішення забезпечує необхідну функціональність, включаючи офлайн-зберігання даних, автоматичну синхронізацію з сервером та високу надійність [20].

PouchDB та CouchDB. PouchDB – це бібліотека JavaScript для зберігання даних на клієнтському пристрої, яка дозволяє синхронізувати ці дані з базою даних CouchDB на сервері. Основні переваги PouchDB та CouchDB:

– офлайн-робота. PouchDB дозволяє зберігати дані локально на клієнтському пристрої та синхронізувати їх з CouchDB після відновлення інтернет-з'єднання [21];

– синхронізація даних. Автоматична двостороння синхронізація між PouchDB та CouchDB забезпечує актуальність даних;

– реплікація [22]. Підтримка реплікації даних між різними інстанціями CouchDB, що забезпечує високу надійність та доступність даних.

Висновки до розділу 2

У другому розділі висвітлені основні технології, які будуть використані для розробки вебзастосунку анкетування з можливістю функціювання в офлайн режимі, а також методи забезпечення цього офлайн-функціоналу.

Для створення вебзастосунку було обрано кілька сучасних технологій. React, як популярна бібліотека JavaScript, дозволяє створювати швидкі та інтерактивні користувацькі інтерфейси завдяки компонентному підходу, використанню віртуального DOM та можливостям управління станом. Redux забезпечує централізоване управління станом, роблячи поведінку застосунку передбачуваною та спрощуючи його відлагодження. React Query надає потужні інструменти для управління даними сервера, забезпечуючи автоматичне оновлення даних, кешування та синхронізацію в режимі реального часу. Node.js з Express обрано для розробки серверної частини завдяки їхній продуктивності та можливості одночасної обробки багатьох запитів. JWT використовується для створення токенів доступу, які забезпечують безпечну аутентифікацію та авторизацію користувачів.

Забезпечення офлайн-функціоналу є ключовим елементом для даного вебзастосунку, особливо в умовах обмеженого або відсутнього інтернет-з'єднання. Офлайн-функціонал дозволяє зберігати дані локально на пристрої користувача та синхронізувати їх з сервером, коли з'явиться інтернет-з'єднання. Найбільш підходящим варіантом для реалізації офлайн-роботи є використання PouchDB у поєднанні з CouchDB. Це рішення забезпечує необхідну функціональність, включаючи офлайн-зберігання даних, автоматичну двосторонню синхронізацію з сервером та високу надійність.

Таким чином, використання перелічених технологій забезпечить створення надійного, ефективного та масштабованого вебзастосунку для анкетування з можливістю роботи в офлайн режимі. Це дозволить задовольнити потреби користувачів та забезпечити ефективний збір і обробку даних навіть в умовах обмеженого або відсутнього інтернет-з'єднання.

3 СТВОРННЯ МАКЕТУ, ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ АНКЕТУВАННЯ ТА ТЕСТУВАННЯ

3.1 Розробка дизайну вебзастосунку у редакторі Figma

Створення проекту було вирішено розпочати із розробки макету вебзастосунку. Для цього було використано платформу Figma [23]. Спочатку був створений макет головної сторінки (рис. 3.1) на якій описується, що представляє із себе застосунок Data Gather та якими можливостями він володіє.

Рисунок 3.1 – Вигляд першої частини макету головної сторінки

На першій частині головної сторінки можна побачити назву застосунку, а також логотипи гуманітарних організацій які, гіпотетично, можуть використовувати даний застосунок. На реальному вебзастосунку планується анімувати логотипи організацій. Зверху розміщена навігаційна панель за допомогою якої можна потрапити на сторінки «Про нас», «Зв’язатися з нами», а також зайти у свій обліковий запис або його створити.

Друга частина головної сторінки (рис. 3.2) розповідає про ключові особливості застосунку та його переваги.

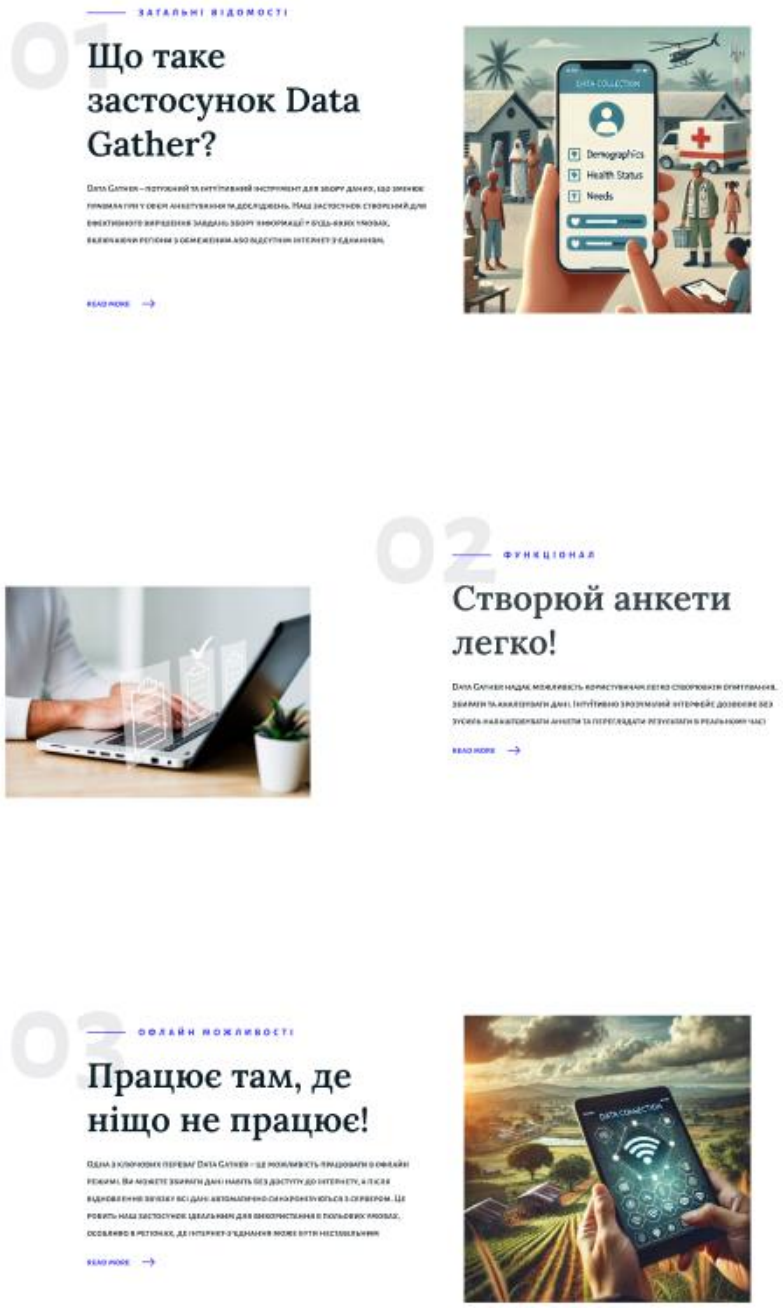

Рисунок 3.2 – Вигляд другої частини макету головної сторінки

Далі був розроблений макет сторінки «Про нас» (рис. 3.3)

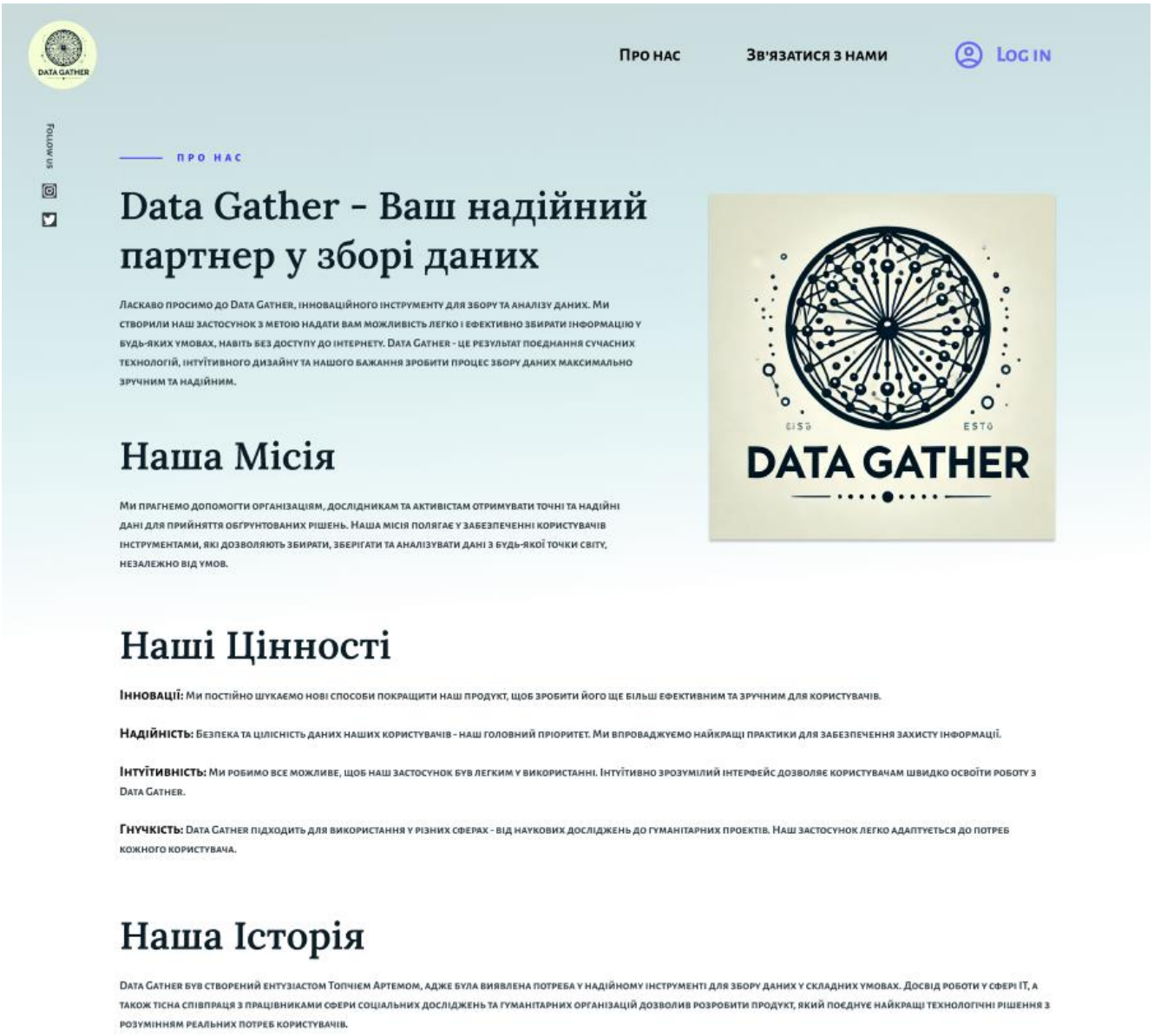

Рисунок 3.3 – Вигляд макету сторінки «Про нас»

Після цього був розроблений макет сторінки «Зв’язатися з нами» (рис. 3.4)

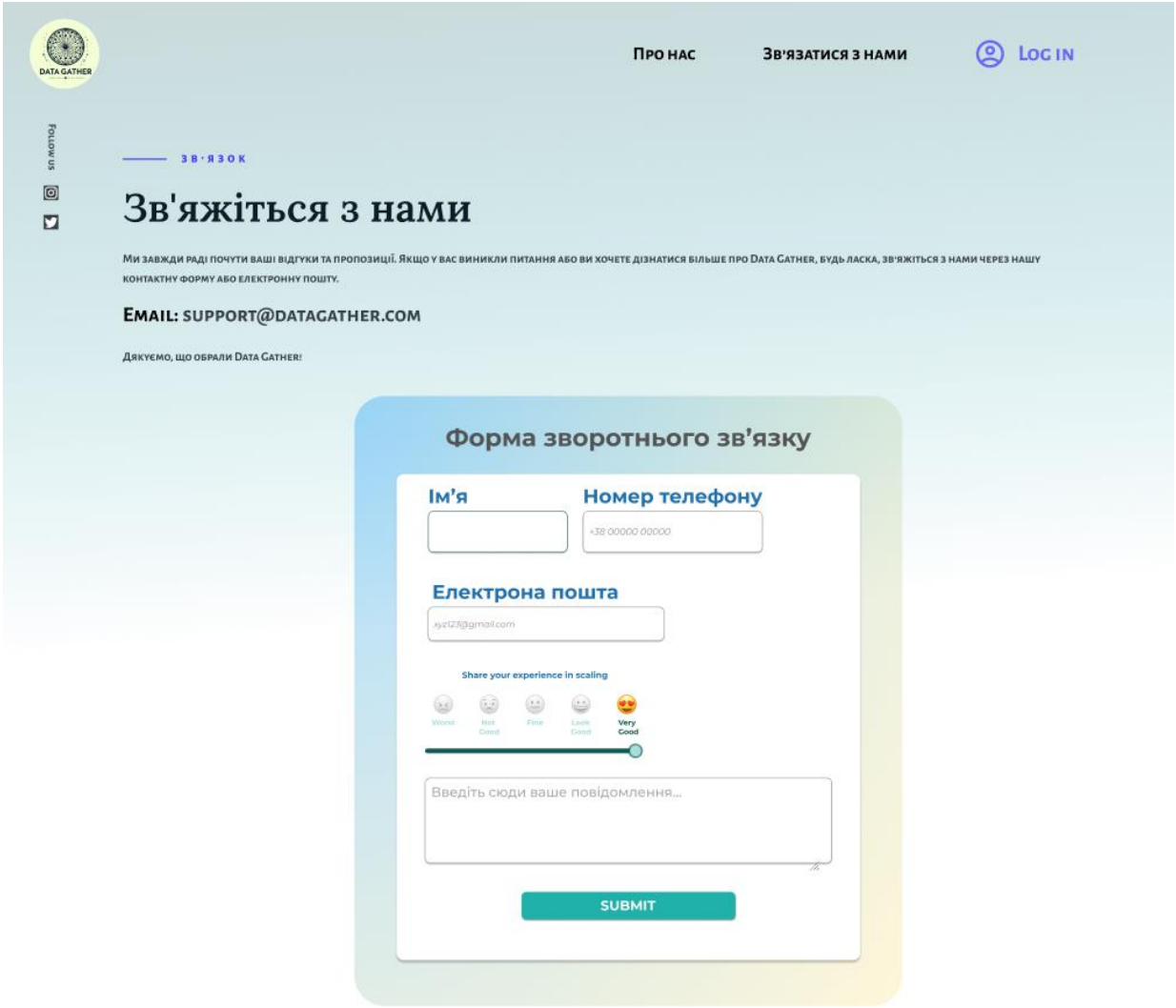

Рисунок 3.4 – Вигляд макету сторінки «Зв’язатися з нами»

На сторінці, окрім контактних даних, була розміщена форма для зворотнього зв’язку.

Всі інші сторінки було вирішено моделювати у ході розробки.

3.2 Опис програмної реалізації

Створення проекту було вирішено розпочати із фронтенд частини. Тому, був створений новий проект на React за допомогою командного рядка та відповідної команди. Відразу він був і запущений, для тестування (рис. 3.5).

```
npx create-react-app
```

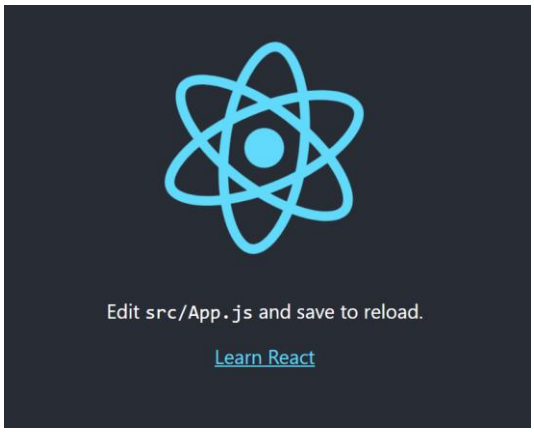

Рисунок 3.5 – Вигляд пустого застосунку після його запуску

Базова структура проекту, створена за замовчуванням, показана на нижче (рис. 3.6).

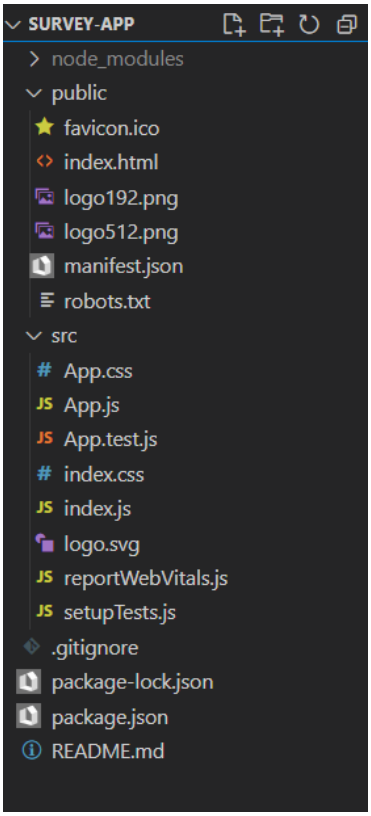

Рисунок 3.6 – Вигляд базової структури проекту

Відразу ж після цього за допомогою пакетного менеджера npm було встановлено декілька додаткових модулів, які знадобляться у майбутньому, у тому числі Redux, React-router для налаштувань маршрутизації, а також база даних PouchDB [24].

```
npm install redux react-redux react-router-dom pouchdb
```

Отже, було розпочато верстання наступних сторінок:

– **головна**. Ця сторінка буде відкриватися за замовчуванням при запуску застосунку;

– **про нас**. На цій сторінці розповідається про те, що являє із себе застосунок Data Gather;

– **зв'язатися із нами**. Ця сторінка дає можливість користувачам залишити відгуки або пропозиції щодо продукту. Зробити це можна або через пошту, або через форму;

– **увійти**. Ця сторінка дозволяє користувачам увійти в існуючий обліковий запис, заповнивши поля «Username» та «Password»;

– **створити аккаунт**. Ця сторінка дозволяє користувачам створити новий обліковий запис, заповнивши поля «Username» та «Password». Після реєстрації користувач відразу перенаправляється до сторінки входу, щоб можна було відразу ж увійти в щойно створений обліковий запис;

– **панель управління**. Це сторінка призначена для перегляду та аналізу результатів, отриманих після проходження людьми опитувань. Також звідси можна потрапити на сторінку для створення нових опитувань. На дану сторінку та всі, на які потрапити можливо лише через цю, можна отримати доступ тільки маючи аккаунт. Варто зауважити, що ця сторінка не потрібна для респондентів, адже проходження опитувань, як вже зазначалося раніше, доступне без реєстрації;

– **сторінка для створення опитувань**. На цю сторінку можна буде потрапляти зі панелі управління, маючи аккаунт. Вона призначена для створення опитувань;

– **сторінка із опитуванням**. На цю сторінку можна буде потрапляти будь-яким способом, у тому числі, маючи пряме посилання на неї, а проходити опитування можна не маючи аккаунту.

Для верстки на цьому етапі було створено декілька компонентів: Home, About, ContactUs, Login, Register, Dashboard, CreateSurveyForm. Також було створено окремий компонент для кнопки, яка повертає користувача на головну сторінку із будь-якої сторінки, де розміщений даний компонент.

На цьому етапі також було вирішено згенерувати логотип сайту (рис. 3.7) за допомогою ШІ DALL-E 3 – нейронної моделі перетворення тексту на зображення, розробленою OpenAI з використанням методологій глибокого навчання для створення цифрових зображень з описів природною мовою [25].

Рисунок 3.7 – Вигляд логотипу вебзастосунку, згенерованого ШІ

Детальніше щодо верстки головної сторінки Home. Сторінка умовно була поділена на 3 частини: верхня частина (хедер) (рис. 3.8), основна частина з контентом, а також нижня частина (футер). У верхній частині було розміщено навігаційну панель у вигляді кнопок з правого боку, а також логотип сайту з лівого.

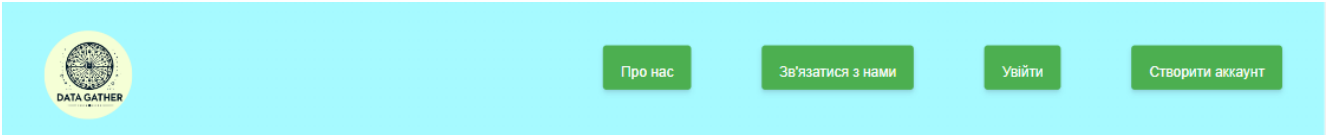

Рисунок 3.8 – Вигляд хедеру

На даний час більшості кнопок був встановлений один стиль. Кути у логотипу були закруглені програмним чином (рис. 3.9).

```
.logo img {
  max-height: 100px;
  border-radius: 100%;
}
```

Рисунок 3.9 – Стиль для закруглення кутів та встановлення розміру логотипу

В якості фону для всіх сторінок був встановлений градієнт (рис. 3.10) – від світло-синього зверху до майже білого знизу.

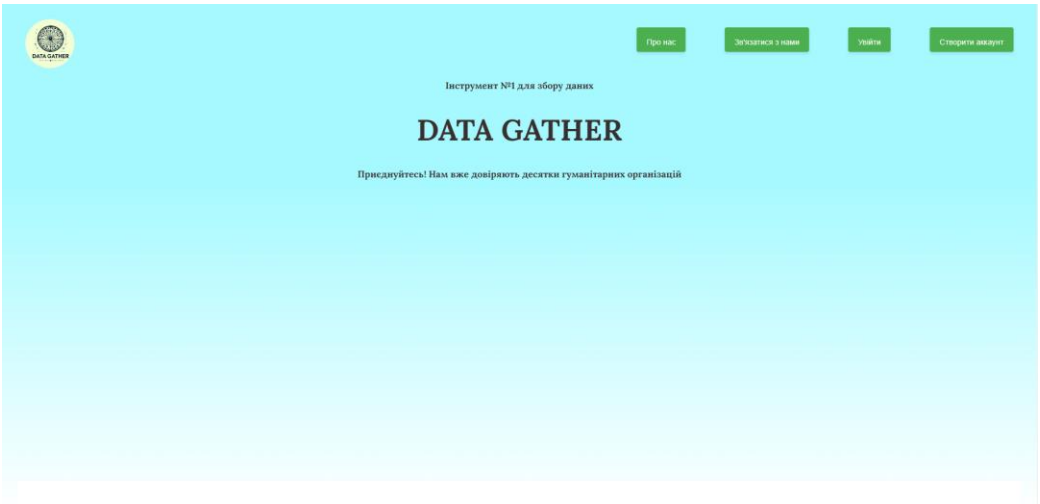

Рисунок 3.10 – Вигляд зверстаної головної сторінки із градієнтом

Структура застосунку побудована так, що весь контент кожної сторінки знаходиться всередині div із відповідним класом. При цьому, у стилях, цьому класу присвоюється градієнт (рис. 3.11).

```
.main {
  display: flex;
  background: rgb(166,250,255);
  background: linear-gradient(180deg, rgba(166,250,255,1) 0%, rgba(166,250,255,1) 35%, rgba(255,255,255,1) 100%);
}
```

Рисунок 3.11 – CSS стилі за допомогою яких задається градієнт

Нижче назви застосунка було додано логотипи гуманітарних організацій (рис. 3.12). При цьому, було вирішено їх анімувати за допомогою бібліотеки **anime.js.** Anime.js - це легка бібліотека анімації JavaScript із простим, але потужним API.

Він працює з властивостями CSS, SVG, атрибутами DOM і об’єктами JavaScript. На даний час було використано чорнову варіацію анімації, у якій логотипи з’являються майже у одній точці, а потім «розпливаються» по екрану, не виходячи за його рамки.

Рисунок 3.12 – Вигляд зверстаної головної сторінки із доданими логотипам організацій та їх анімацією

Варто відзначити, що при кожному перезавантаженні сторінки змінюється порядок та напрям руху кожного з логотипів. Це відбувається завдяки тому, що координати позиції генеруються випадковим чином у функції getRandomPosition.

```
const getRandomPosition = (minX, maxX, minY, maxY) => {
  const x = Math.random() * (maxX - minX) + minX;
  const y = Math.random() * (maxY - minY) + minY;
  return { x, y };
};
```

А не вихід за рамки екрану та сама анімація відбувається за допомогою хуку useEffect.

```
const logos = document.querySelectorAll(".logo-ngo");
    const minX = window.innerWidth * -0.2;
    const maxX = window.innerWidth * 0.12;
    const minY = window.innerHeight * 0;
    const maxY = window.innerHeight * 0.2;

    logos.forEach((logo) => {
      const { x, y } = getRandomPosition(minX, maxX, minY, maxY);
      logo.style.transform = `translate(${x}px, ${y}px)`;
```

Детальний код анімації наведений у ДОДАТКУ А.

Після логотипів іде «рекламна» частина сайту із кричущими назвами (рис. 3.13).

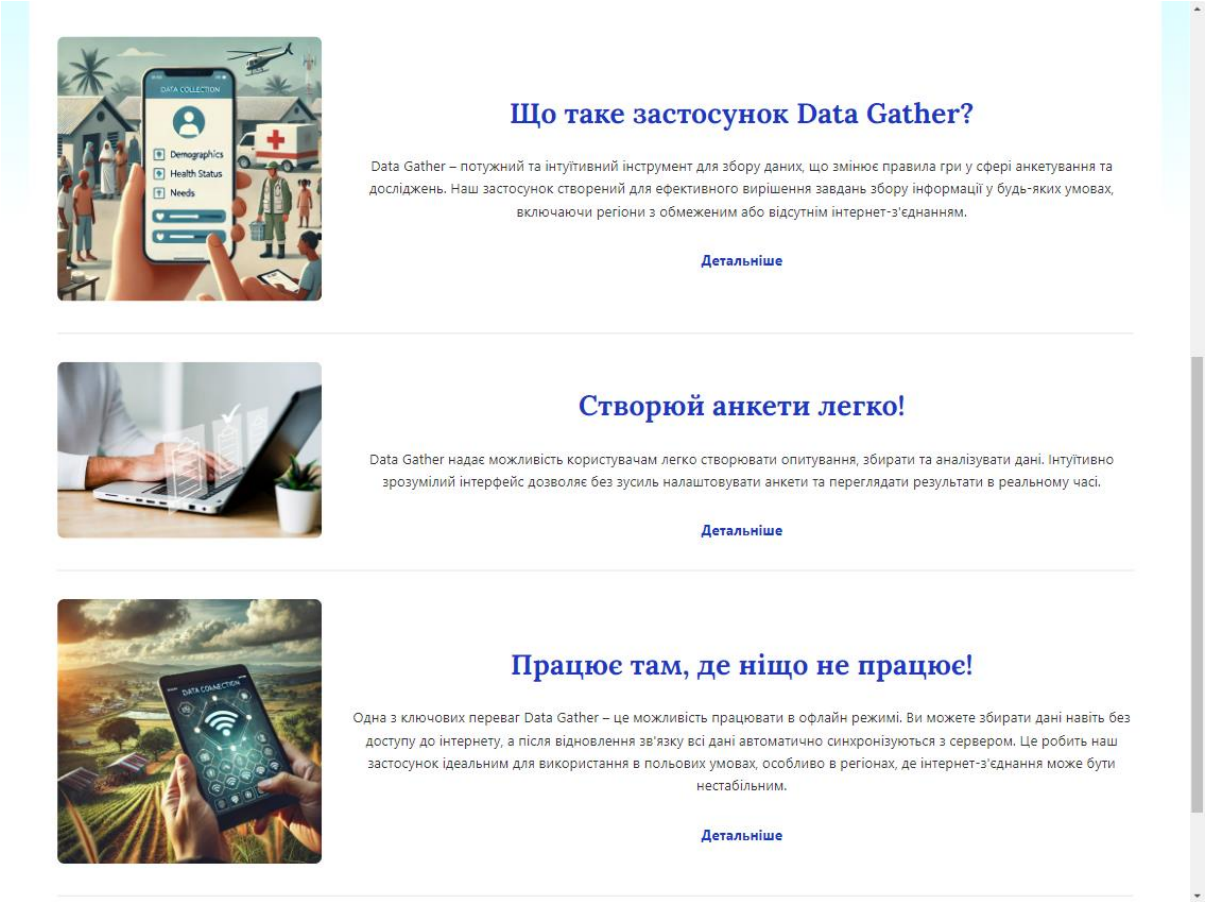

Рисунок 3.13 – Основна зверстана частина головної сторінки

Знизу ж доданий звичайний статичний футер (рис. 3.14).

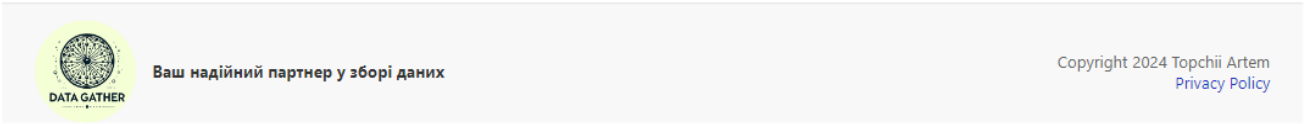

Рисунок 3.14 – Футер

Наступні сторінки зроблені подібним чином та з подібними стилями, наприклад, сторінка «Про нас» (рис. 3.15).

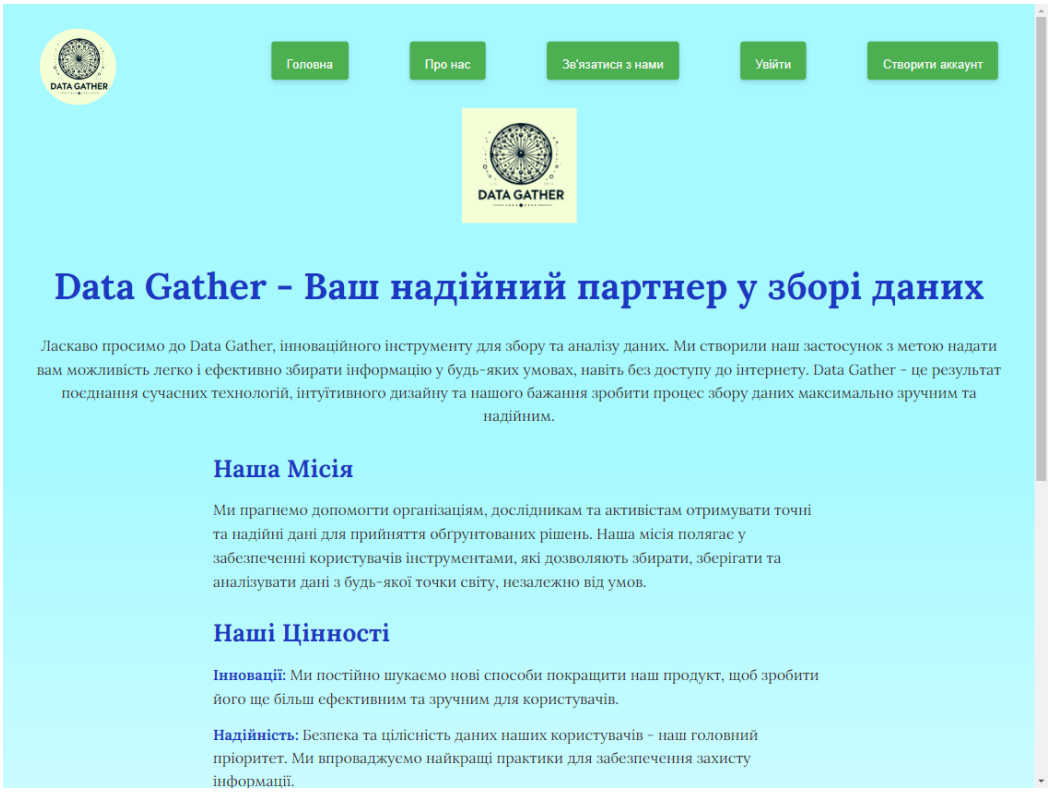

Рисунок 3.15 – Зверстана сторінка «Про нас»

На сторінці «Зв’язатися з нами» (рис. 3.16) була зверстана проста форма зворотнього зв’язку, яка, за задумом, після натискання кнопки «Відправити» записує дані у локальний текстовий файл 'contact_messages.txt' в директорії проекту.

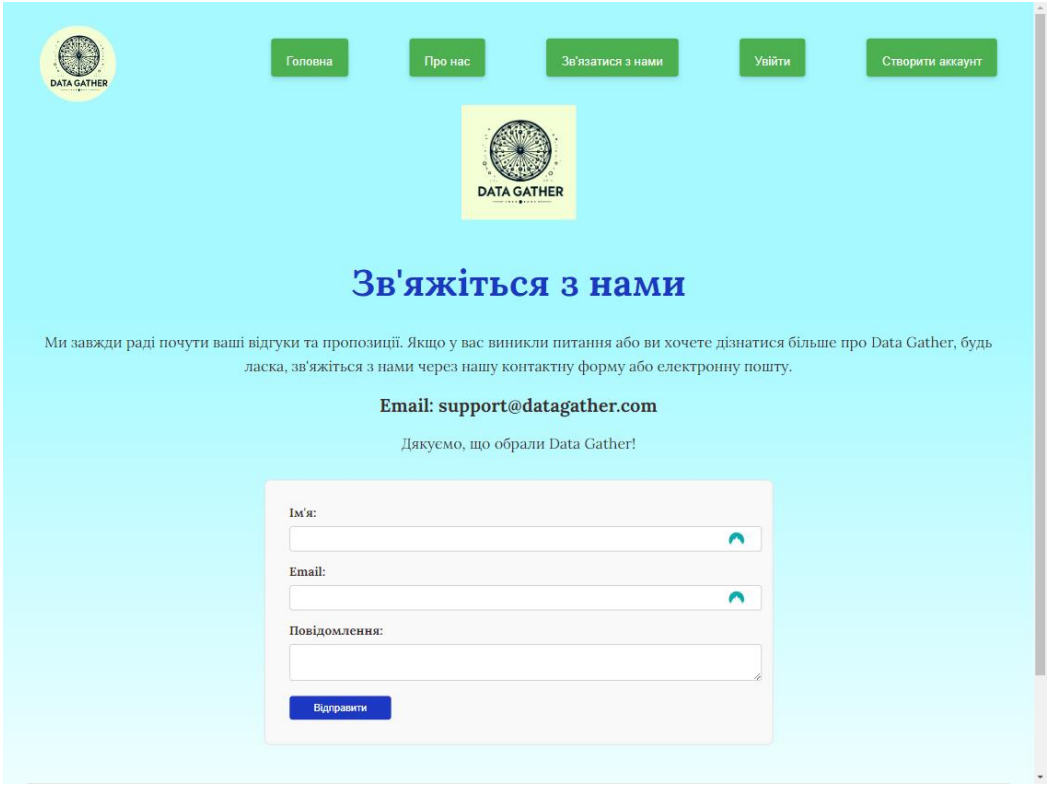

Рисунок 3.16 – Зверстана сторінка «Зв’язатися з нами»

Оскільки React працює на стороні клієнта, неможливо безпосередньо зберігати файли на локальний диск з клієнтської частини. Для реалізації такої функціональності, була використана серверна частина для зберігання файлів – на основі серверу Node.js та Express. Це дозволить обробляти форму та зберігати дані у файл.

Було створено файл server.js у корені проекту. Після цього було встановлено необхідні npm пакети для забезпечення роботи серверу та процесу зберігання у файл, а саме: **express, body-parser** та **cors.**

```
npm install express body-parser cors
```

Express.js - це мінімалістичний веб-фреймворк для Node.js, який спрощує розробку веб-застосунків та API. Express надає набір інструментів для обробки HTTP-запитів, маршрутизації, налаштування middleware та багато іншого.

Body-parser - це middleware для Express.js, яке допомагає аналізувати (парсити) тіло HTTP-запиту. Воно перетворює тіло запиту у зручний формат, такий

як JSON, щоб ви могли легко отримувати доступ до даних, переданих у запиті. В даному застосунку він використовується для парсингу JSON-даних, відправлених клієнтом у тілі запиту та обробки даних форм, відправлених як application/x-www-form-urlencoded.

CORS - це механізм безпеки браузера, який дозволяє веб-застосункам на одному домені робити запити до ресурсів на іншому домені. За замовчуванням, браузери блокують такі запити через політику однакового походження (same-origin policy). В даному застосунку було увімкнено обробку CORS на сервері, щоб дозволити запити з клієнтського застосунку. Частину коду, який оброблює дані з форми, можна побачити нижче.

```
app.post('/api/contact', (req, res) => {
  const { name, email, message } = req.body;
  const logMessage = `Name: ${name}\nEmail: ${email}\nMessage: ${message}\n\n`;

  fs.appendFile(path.join(__dirname, 'contact_messages.txt'), logMessage, (err) =>
{
    if (err) {
      console.error('Error saving feedback:', err);
      return res.status(500).send('Server error');
    }
    res.send('Message received');
  });
});
```

У цьому коді за допомогою **POST** запиту дані відправляються на сервер, а за допомогою модулю Node.js під назвою fs та його методу **appendFile** дані записуються у кінець текстового файлу.

Загалом, процес обробки даних з форми фідбеку виглядає наступним чином:

– **отримання даних із форми**: Коли користувач відправляє форму зворотного зв'язку, дані (ім'я, електронна адреса та повідомлення) відправляються на сервер;

– **формування повідомлення**: Сервер формує текстове повідомлення logMessage, яке містить отримані дані у форматі, придатному для запису у файл;

– **додавання даних до файлу**: Використовуючи метод fs.appendFile, сервер додає сформоване повідомлення до файлу contact_messages.txt. Якщо файл не існує, він буде створений;

– **обробка помилок**: Якщо під час додавання даних до файлу виникає помилка, вона обробляється і виводиться в консоль. Користувачу відправляється відповідь про помилку сервера. Якщо все пройшло успішно, користувачу відправляється повідомлення про успішне отримання повідомлення.

Від цього часу для функціонування проекту потрібно запускати завжди і сам проект, і node сервер за допомогою команди.

```
node server.js
```

Далі були зверстані сторінки «Створити аккаунт» (рис. 3.17) та «Увійти» (рис. 3.18).

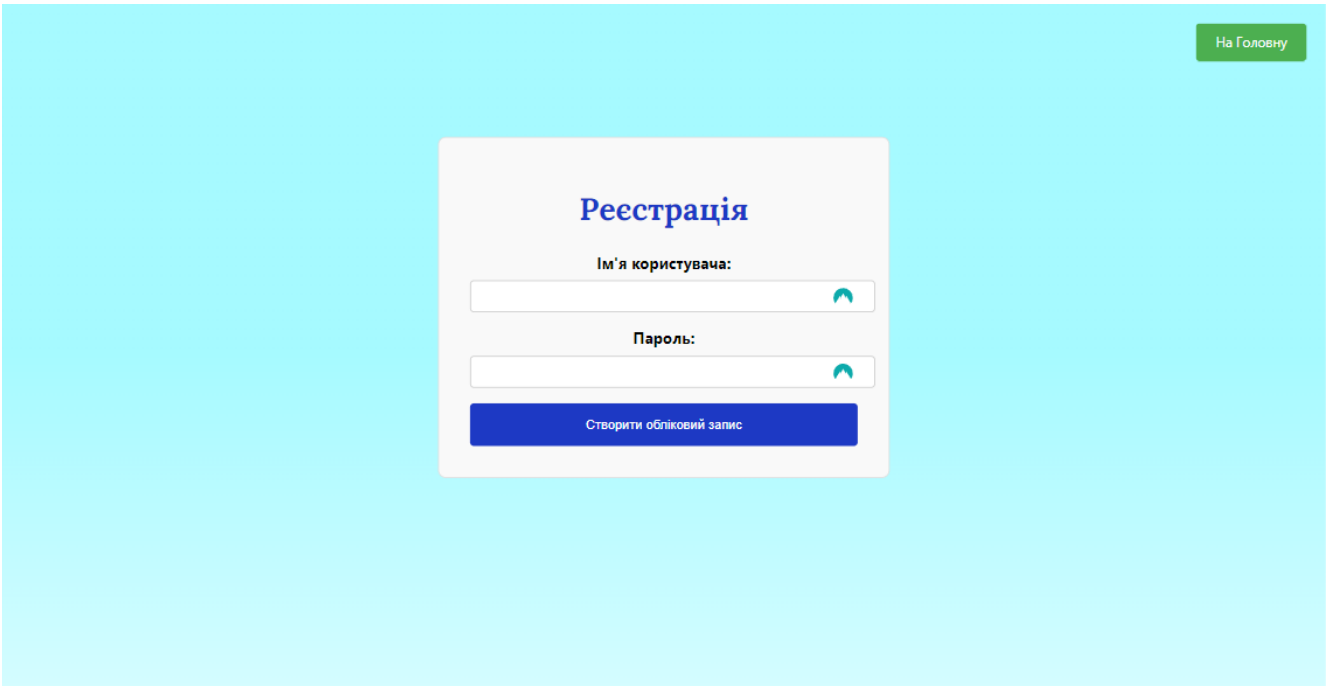

Рисунок 3.17 – Зверстана сторінка «Створити аккаунт»

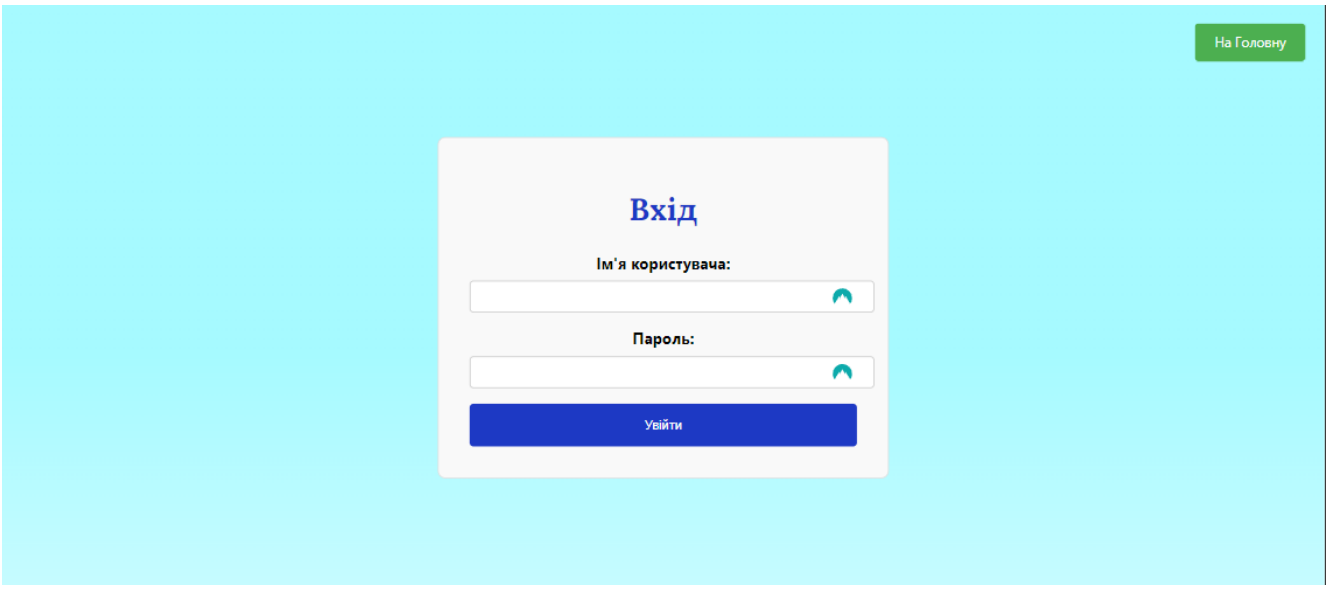

Рисунок 3.18 – Зверстана сторінка «Увійти»

Через те, що немає вимоги розроблювати офлайн авторизацію, адже в цьому немає сенсу, було остаточно вирішено використовувати базу даних CouchDB для процесу авторизації.

Для реалізації процесу авторизації були виконані наступні 3 пункти:

– налаштування CouchDB для зберігання користувачів;
– реалізація серверної логіки для реєстрації та входу користувачів;
– реалізація клієнтської частини для взаємодії з сервером.

Спочатку було встановлено CouchDB з офіційного сайту. При встановленні, було відразу створено користувача admin із всіма правами доступу. Далі за допомогою переходу по посиланню http://localhost:5984/_utils/ було запущено Fauxton (рис. 3.19) - це веб-інтерфейс адміністрування CouchDB та було виконано вхід за допомогою облікових даних, які вказувалися при встановленні CouchDB.

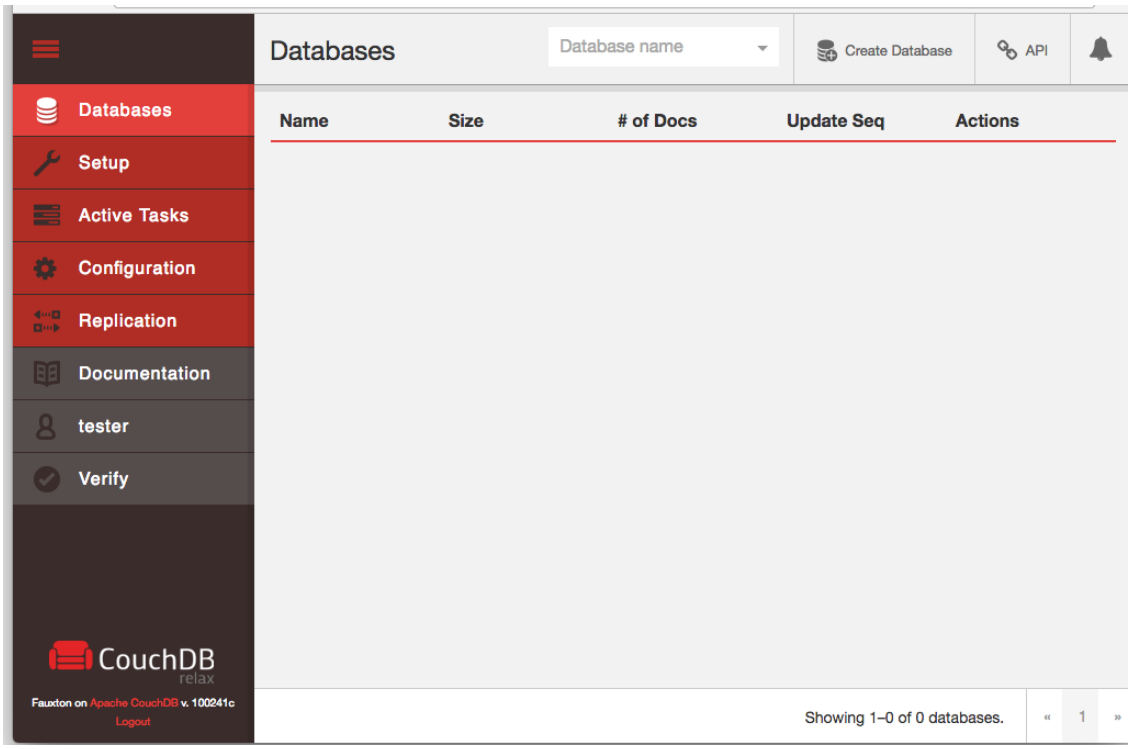

Рисунок 3.19 – Вигляд веб-інтерфейс Fauxton

За допомогою кнопки «Create Database» було створено базу даних users (рис. 3.20).

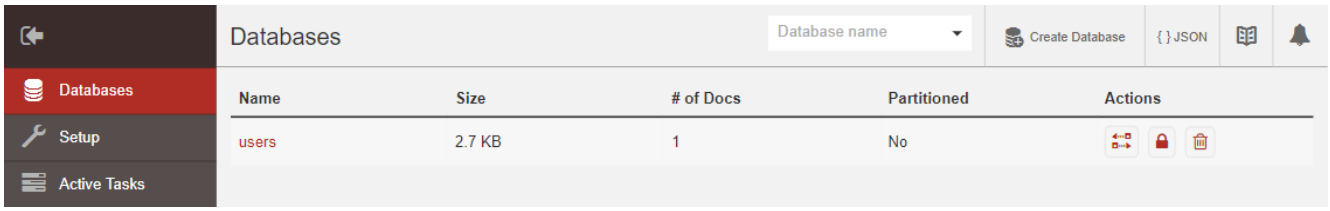

Рисунок 3.20 – База даних users

Для того, щоб забезпечити безпеку даних, база даних CouchDB має так звані «ролі», які можна встановити користувачам. Ці ролі впливають на те, якій рівень доступу буде у користувача. Неавторизовані користувачі не повинні мати доступу до панелі керування та бачити результатів опитування. При цьому, авторизовані користувачі повинні мати можливість створити власне опитування, а також передивлятися результати опитувань, які були створені з поточного авторизованого аккаунту користувача. Для цього, було створено дизайн-документ у базі даних CouchDB, в якому вказується, що при створенні користувача, йому автоматично

присвоюється роль «member» завдяки якій користувач буде мати всі необхідні доступи.

```
{
  "_id": "_design/users",
  "_rev": "2-ba9f7ecc49af5c4e593c2e8538b8c778",
  "views": {
    "by_username": {
      "map": "function (doc) { emit(doc.username, doc); }"
    },
    "by_role": {
      "map": "function (doc) { if(doc.roles) { doc.roles.forEach(role => emit(role,
doc)); } }"
    }
  }
```

Після закінчення створення бази даних було оновлено серверну частину, а саме, файл server.js. Фрагмент оновленого коду, який відповідає за реєстрацію нових користувачів, можна побачити нижче.

```
app.post('/api/register', async (req, res) => {
  const { username, password } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const user = await db.insert({
      username,
      password: hashedPassword,
      roles: ['member']
    });
    console.log('User registered successfully:', user);
    res.status(201).send('User registered successfully');
  } catch (err) {
    console.error('Error registering user:', err);
    res.status(500).send('Error registering user');
  }
```

Реєстрація працює таким чином, що коли клієнт натискає на кнопку «Створити обліковий запис» після заповнення форми, автоматично надсилається POST-запит на /api/register, далі цей маршрут обробляється функцією, яка починається з app.post('/api/register', async (req, res) => { ... }).

Дані з форми реєстрації (username і password) передаються в тілі запиту (req.body). Константи username і password отримуються з тіла запиту за допомогою деструктуризації: const { username, password } = req.body;. Пароль користувача

хешується перед збереженням у базі даних для забезпечення безпеки. Це відбувається за допомогою бібліотеки bcrypt.

Метод await bcrypt.hash(password, 10) створює хеш пароля з використанням 10 раундів хешування. Використовується метод insert для додавання нового документа в базу даних users. Документ містить такі поля: username, password (захешований) і roles (список ролей користувача, де користувачеві присвоюється роль member).

await db.insert({ username, password: hashedPassword, roles: ['member'] }); додає нового користувача в базу даних. Якщо користувача успішно зареєстровано, сервер відправляє відповідь з кодом статусу 201 і повідомленням 'User registered successfully'. Якщо виникає помилка під час реєстрації, сервер відправляє відповідь з кодом статусу 500 і повідомленням 'Error registering user', також вона ловиться в блоці catch, і деталі помилки виводяться в консоль. Клієнт отримує відповідь про помилку з кодом статусу 500.

У разі, якщо користувач вже зареєстрований, є можливість увійти в обліковий запис. Зробити це можна за допомогою кнопки «Увійти», після якої користувача перенаправляє на сторінку входу. Вона розроблена схожим чином із сторінкою реєстрації, але використовує інший серверний код, який можна побачити нижче.

```
app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;
  try {
    const response = await db.view('users', 'by_username', { key: username });
    if (response.rows.length === 0) {
      return res.status(400).send('User not found');
    }
    const user = response.rows[0].value;
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      return res.status(400).send('Invalid password');
    }
    const token = jwt.sign({ username: user.username, password }, SECRET_KEY, {
expiresIn: '1h' });
    res.json({ token });
  } catch (err) {  }
});
```

Вхід в існуючий обліковий запис користувача відбувається таким чином, що коли клієнт натискає на кнопку «Увійти» після заповнення форми, автоматично надсилається POST-запит на /api/login. POST-запит — це HTTP-запит, який використовується для відправки даних на сервер для створення або оновлення ресурсу. Цей маршрут обробляється асинхронною функцією, яка починається з app.post('/api/login', async (req, res) => { ... }). Дані з форми входу (username і password) передаються в тілі запиту (req.body). Константи username і password отримуються з тіла запиту за допомогою деструктуризації: const { username, password } = req.body;. Деструктуризація — це синтаксичний прийом в JavaScript, який дозволяє розпаковувати значення з масивів або властивості з об'єктів в окремі змінні.

Сервер виводить у консоль повідомлення про те, що користувач намагається увійти: console.log('Logging in user:', username);. Сервер виконує запит до бази даних для отримання інформації про користувача за його ім'ям: const response = await db.view('users', 'by_username', { key: username });. Якщо користувач не знайдений (масив response.rows порожній), сервер виводить у консоль повідомлення про відсутність користувача і відправляє клієнту відповідь з кодом статусу 400 та

повідомленням 'User not found': console.log('User not found'); return res.status(400).send('User not found');.

Якщо користувач знайдений, сервер отримує об'єкт користувача: const user = response.rows[0].value;. Сервер порівнює введений пароль із захешованим паролем у базі даних за допомогою bcrypt: const isPasswordValid = await bcrypt.compare(password, user.password);. Якщо пароль недійсний, сервер виводить у консоль повідомлення про недійсний пароль і відправляє клієнту відповідь з кодом статусу 400 та повідомленням 'Invalid password': console.log('Invalid password'); return res.status(400).send('Invalid password');.

Якщо пароль правильний, сервер створює токен JWT, який містить ім'я користувача і пароль, та підписується за допомогою секретного ключа, строк дії якого встановлюється на 1 годину: const token = jwt.sign({ username: user.username, password }, SECRET_KEY, { expiresIn: '1h' });. JWT (JSON Web Token) — це компактний і самодостатній спосіб передачі інформації між сторонами у вигляді JSON-об'єкта. Він може бути використаний для аутентифікації і авторизації. Сервер виводить у консоль повідомлення про успішний вхід користувача: console.log('User logged in successfully:', username);. Сервер відправляє клієнту токен у відповідь: res.json({ token });. Токен JWT, створений при вході, має строк дії 1 годину. Це означає, що через 1 годину токен стане недійсним, і користувачеві потрібно буде знову увійти в систему, щоб отримати новий токен. Це робиться для забезпечення безпеки, щоб мінімізувати ризик компрометації облікових даних.

Після успішного входу токен зберігається у локальному сховищі, а після цього користувача перенаправляє на сторінку «Панелі керування», це можна побачити у коді нижче.

```
    if (response.ok) {
      const data = await response.json();
      localStorage.setItem("token", data.token);
      navigate("/dashboard");
    }
```

Якщо під час процесу входу виникає будь-яка помилка, вона ловиться в блоці catch, і деталі помилки виводяться в консоль: console.error('Error logging in:', err);. Клієнт отримує відповідь про помилку з кодом статусу 500 та повідомленням 'Error logging in': res.status(500).send('Error logging in');.

Наступним кроком була зверстана панель керування (рис. 3.21), на яку перенаправляється користувач автоматично після успішної авторизації.

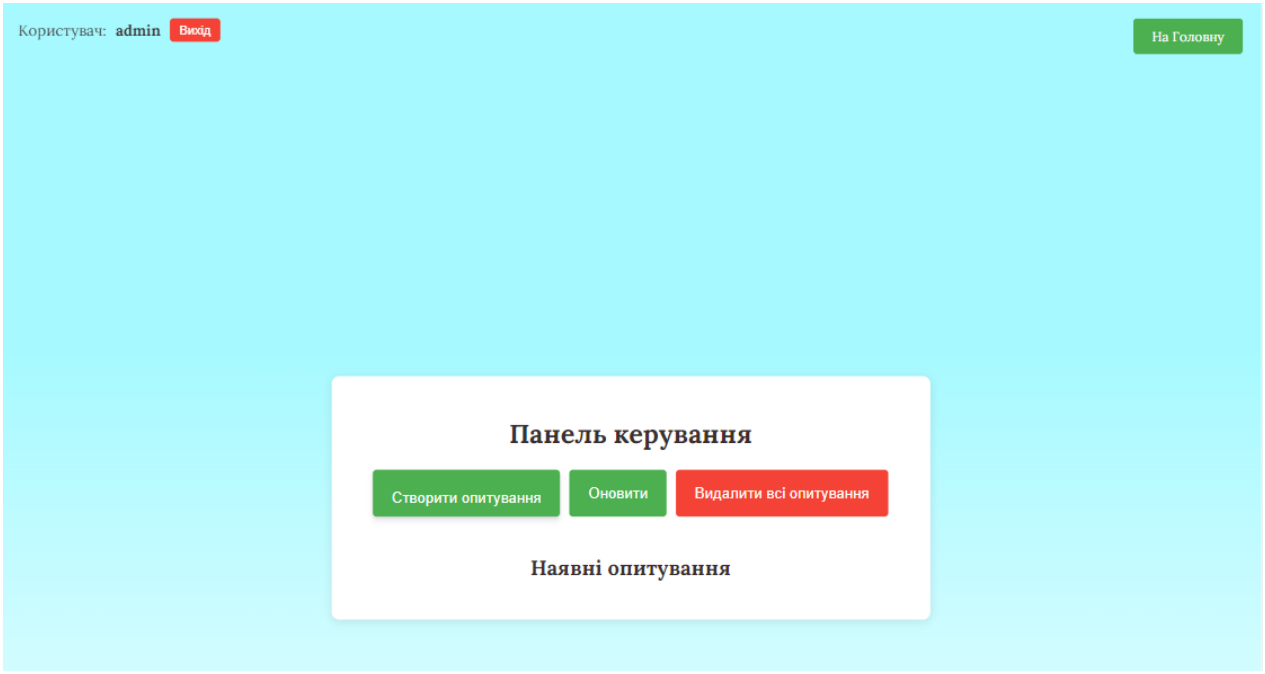

Рисунок 3.21 – Вигляд панелі керування

Панель керування має такі можливості:

– відображення імені поточного користувача;
– можливість виходу з аккаунту;
– можливість переходу назад на головну сторінку;
– можливість створення нового опитування;
– можливість видалення всіх опитувань;
– можливість видаляти окремо кожне з опитувань;
– можливість оновлення списку наявних опитувань;
– можливість перегляду відповідей на кожне з опитувань;
– можливість копіювати публічне посилання на кожне з опитувань.

При натисканні на кнопку «Створити опитування» користувач потрапляє на сторінку створення опитування (рис. 3.22).

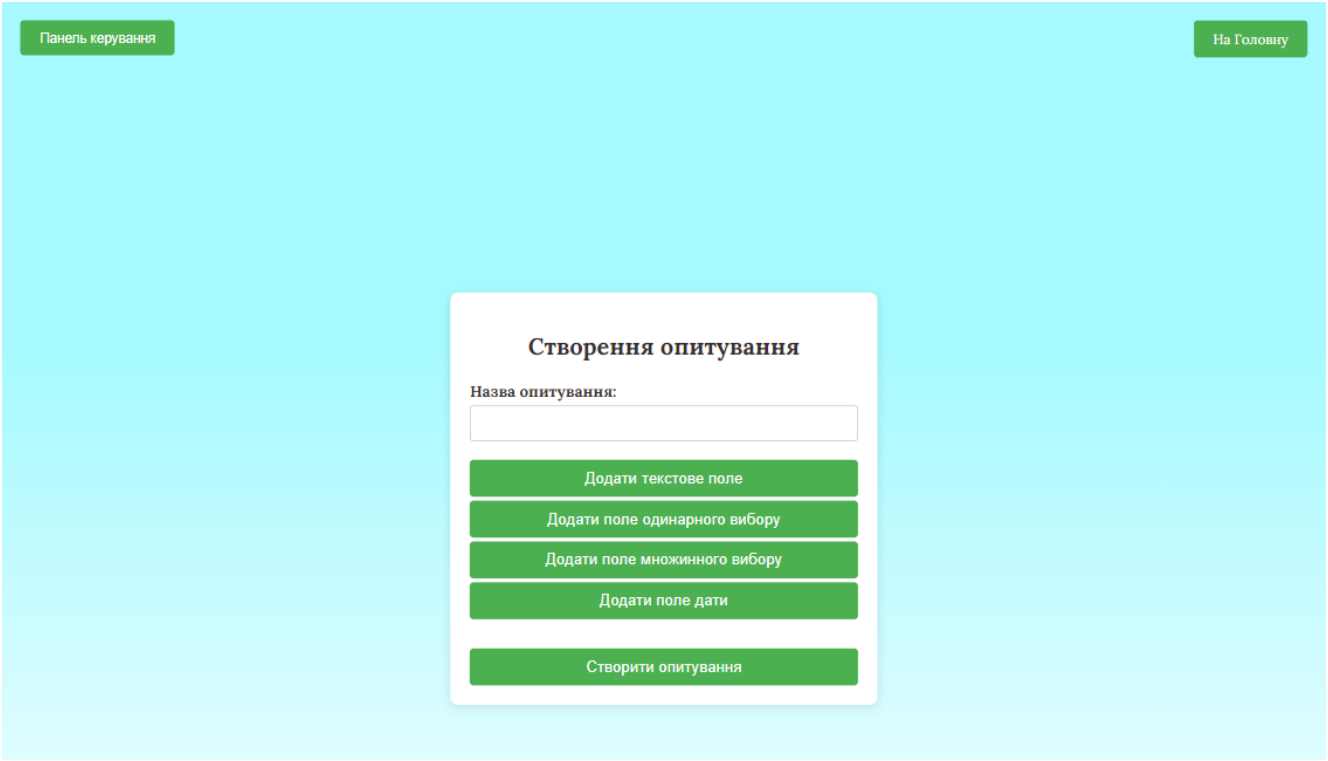

Рисунок 3.22 – Вигляд сторінки створення опитування

На сторінці для створення опитувань можливо виконувати такі дії:

– повернення назад до панелі керування за допомогою кнопки «панель керування», якщо опитування було вирішено не створювати;

– можливість введення назви опитування;

– можливість додавання 4 типів полів, а саме: текстове, поле одинарного вибору, поре множинного вибору, поле для введення дати;

– кнопка для безпосередньо створення опитування.

При мінімальному заповненні форми для створення опитування та натисканні кнопки «Створити опитування», користувач потрапляє на фінальну сторінку, де можна побачити (рис. 3.23), чи було опитування створене і скопіювати публічне посилання на нього для подальшої розсилки.

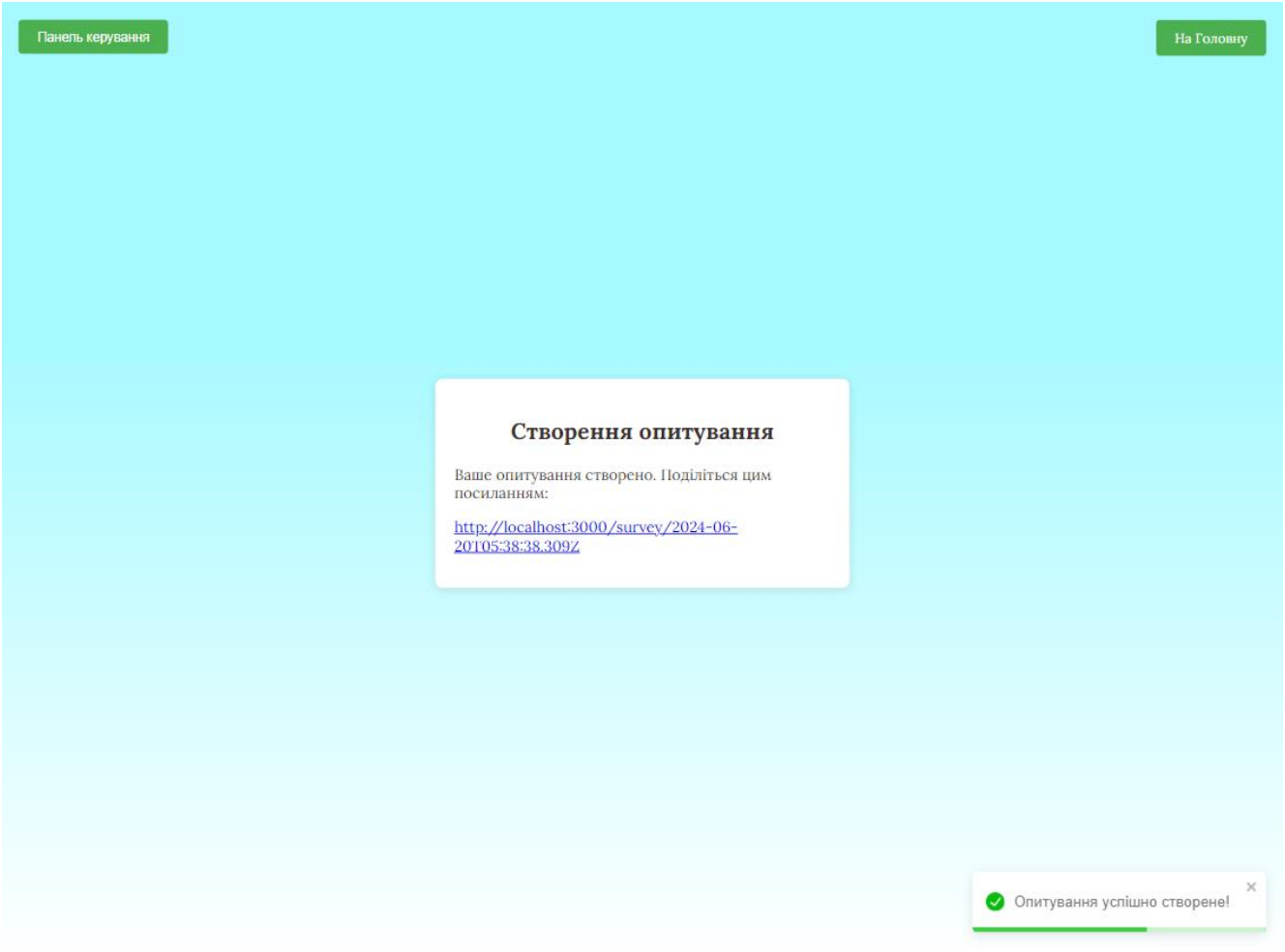

Рисунок 3.23 – Вигляд сторінки з результатом створення опитування

Також на даній сторінці можливо відразу перейти по унікально згенерованому посиланню саме на це опитування. Зробивши це, користувач потрапляє на сторінку для проходження опитування (рис. 3.24).

Рисунок 3.24 – Вигляд сторінки для проходження опитування

Заповнивши опитування та натиснувши кнопку «Відправити», опитування з даними відправляється на сервер.

Перейшовши до панелі керування (рис. 3.25), попередньо авторизувавшись під відповідним користувачем, яким було створене опитування, можна побачити в списку опитування, а також те, що на нього є одна відповідь.

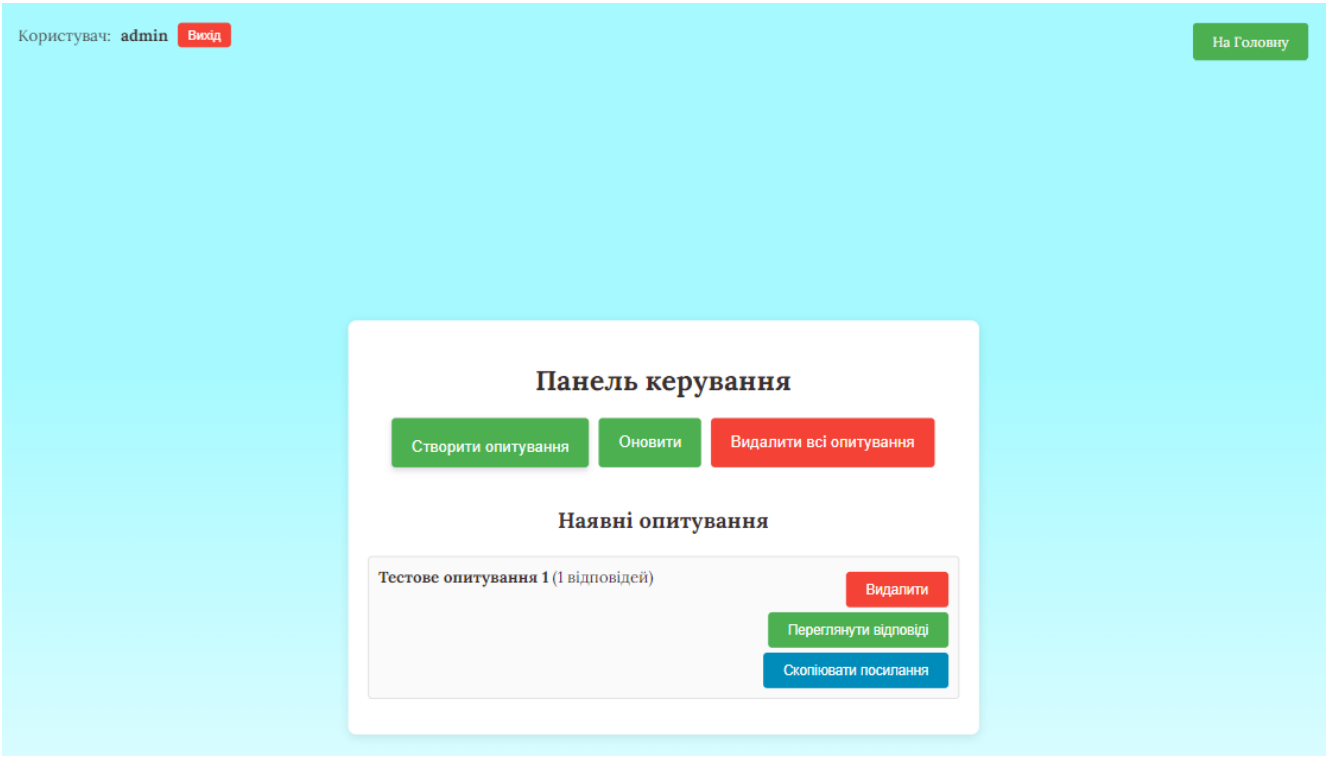

Рисунок 3.25 – Вигляд панелі керування із наявним одним опитуванням

Після цього користувач, яким було створене опитування, може натиснути кнопку «Переглянути відповіді» для перегляду наявних відповідей на це опитування. В такому разі, буде відкрито модальне вікно (рис. 3.26) де можна ознайомитися з результатами проходження людьми опитування. Кожна відповідь поміщується у випадаючий список, також кожна відповідь має свій порядковий номер та точну дату, коли вона була записана до бази даних.

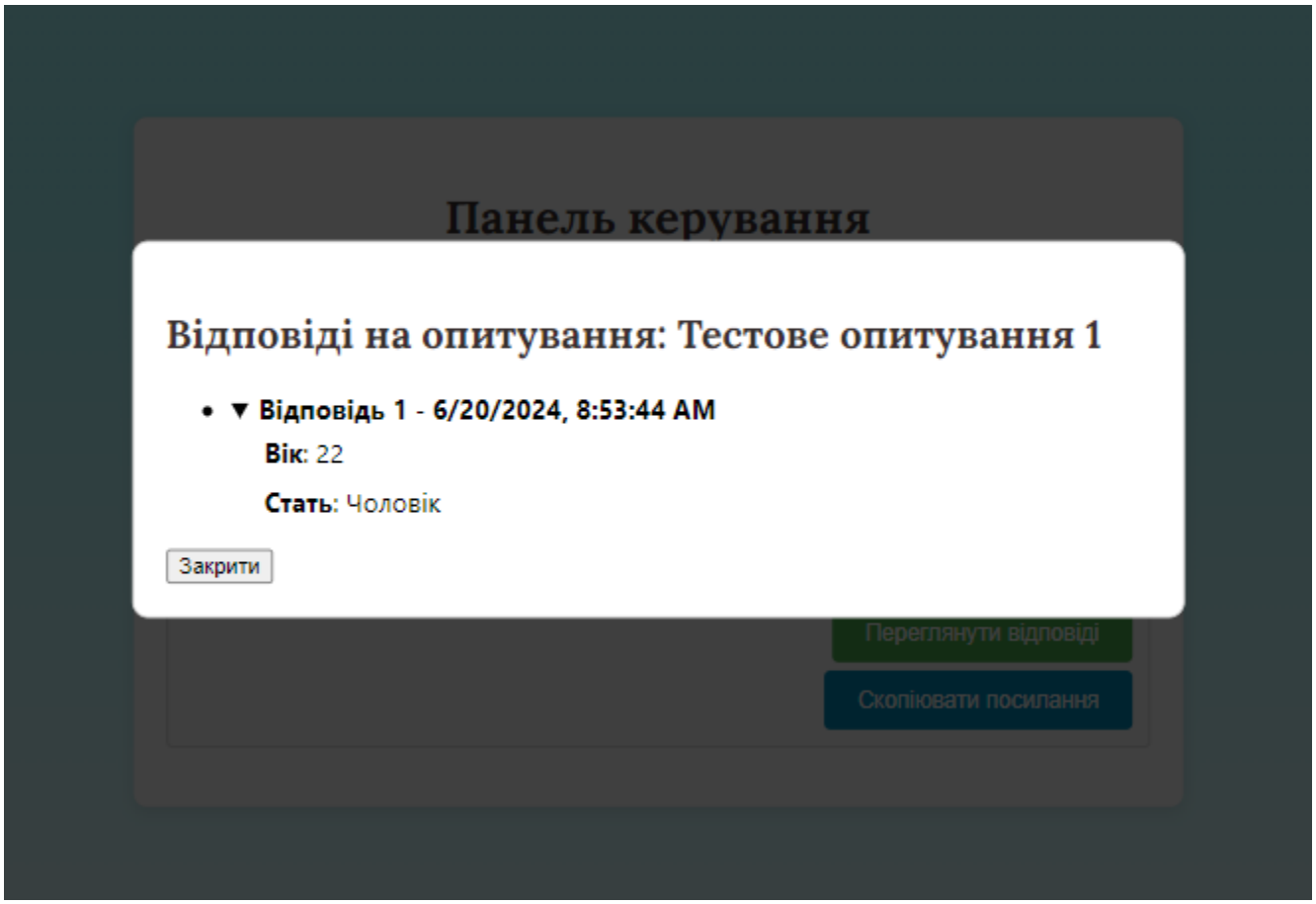

Рисунок 3.26 – Вигляд модального вікна для перегляду результатів проходження опитування

3.2 Тестування веб-застосунку

Оскільки просте створення опитувань та перегляд результатів вже було перевірено в ході виконання роботи, в цьому розділі було вирішено приділити увагу тестуванню безпеки та роботі застосунку в поганих умовах зв’язку, адже у анкетувальних застосунках такого роду це одна з найважливіших складових.

Для початку було створено нового користувача artem1 (рис. 3.27).

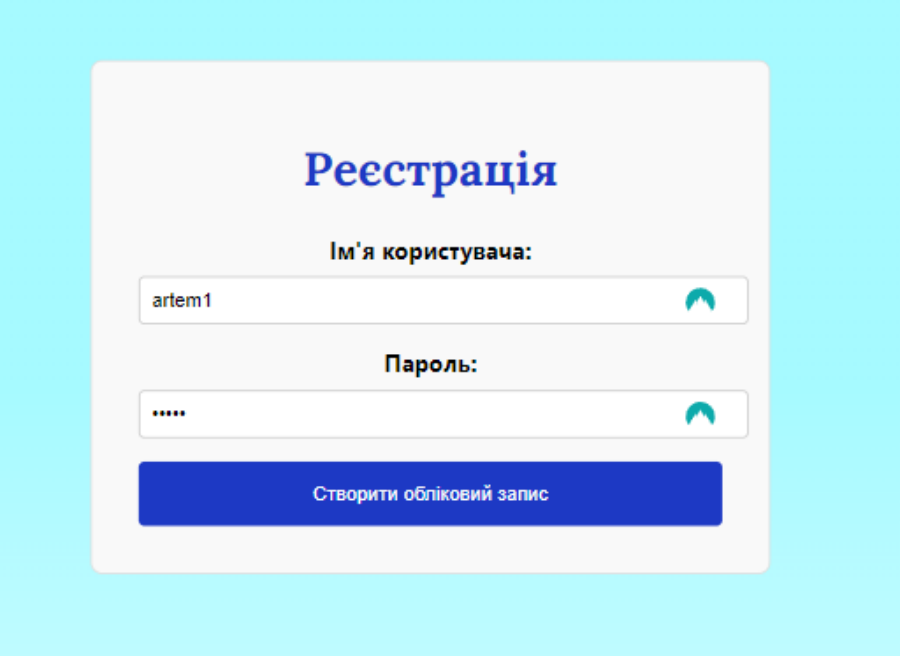

Рисунок 3.27 – Створення нового облікового запису

Після цього було оглянуто панель керування (рис. 3.28). Після аналізу стало зрозуміло, що ніяких опитувань немає, хоча в інших облікових записах вони створювалися. Це означає, що механізм безпеки працює вірно і у кожного користувача свої опитування, які він може видаляти або передивлятися результати, не конфліктуючи при цьому з іншими.

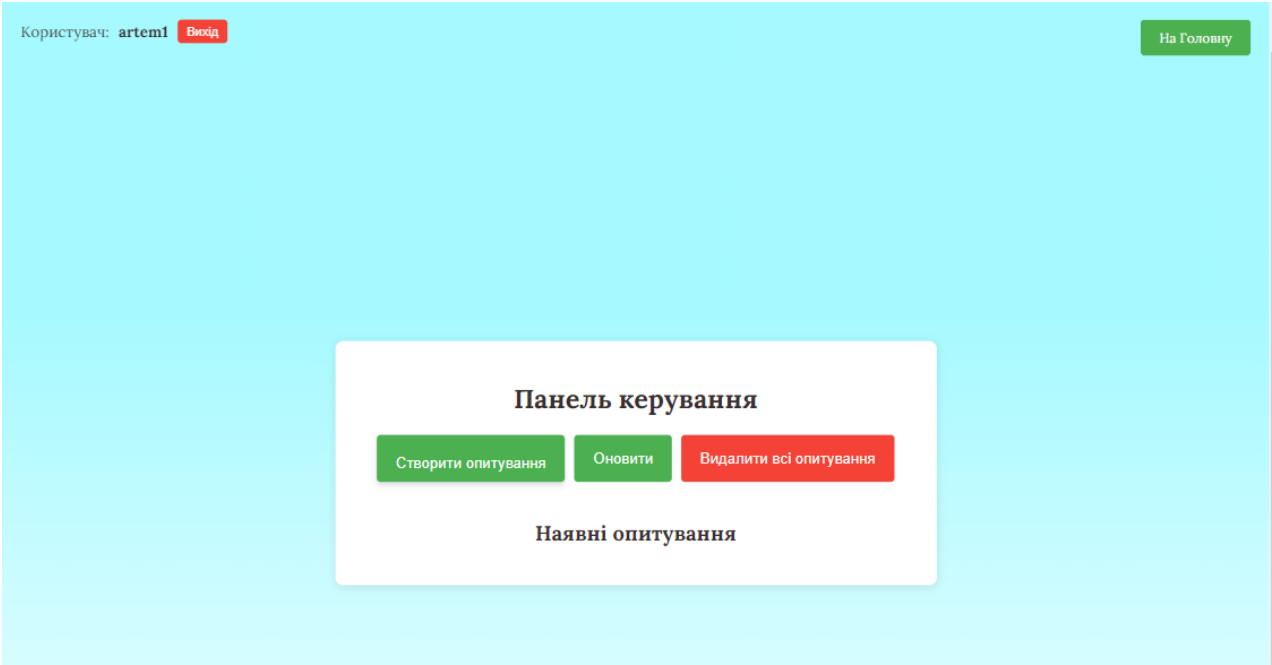

Рисунок 3.28 – Панель керування без «чужих» опитувань

Для додаткової перевірки було додано декілька опитувань (рис. 3.29).

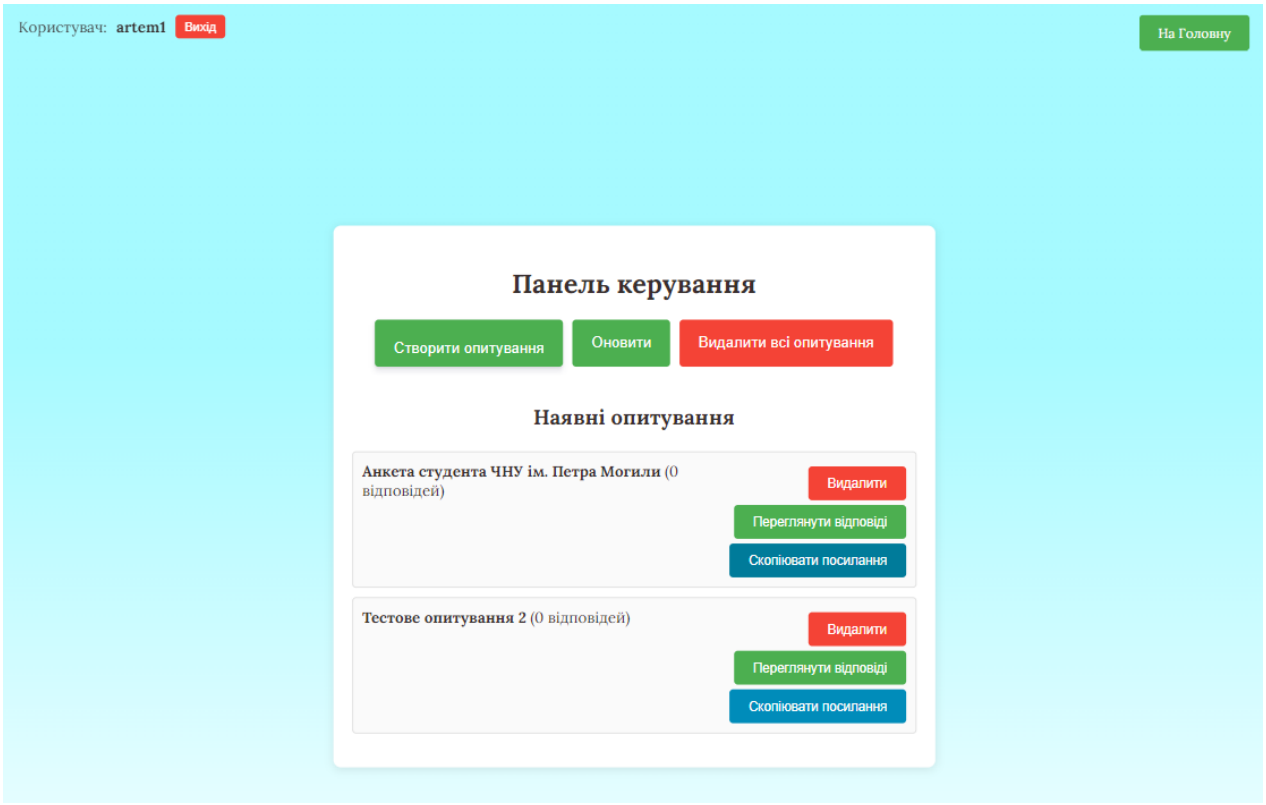

Рисунок 3.29 – Панель керування без «чужих» опитувань

Функціонал додавання опитувань працює без нарікань.

Наступним кроком стало тестування на предмет збереження відповідей під час проходження опитування, у випадку раптового відключення Інтернету. Спочатку було перейдено на сторінку з опитуванням «Анкета студента ЧНУ ім. Петра Могили» із увімкненим інтернетом. Сторінка була завантажена (рис. 3.30).

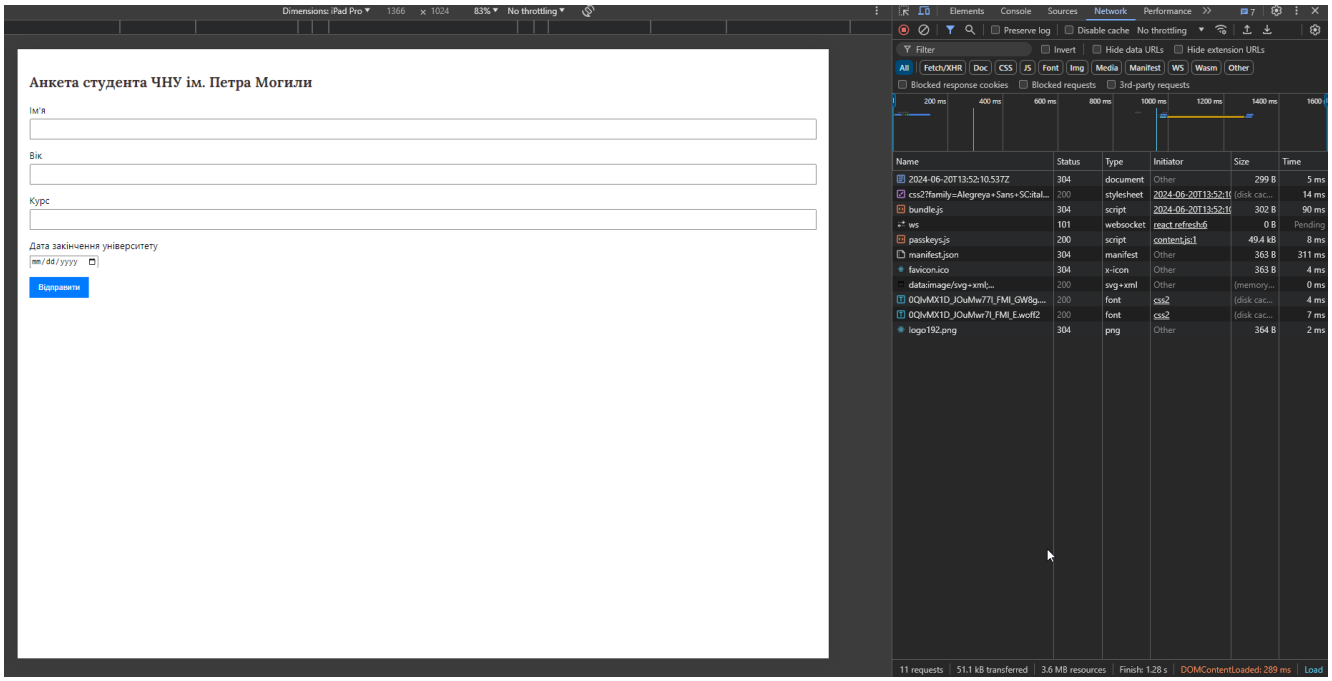

Рисунок 3.30 – Сторінка з опитуванням завантажилася

Для тестування раптового зникання мережі, за допомогою Chrome Dev Tools на вкладці Network було встановлено пресет «Offline» (рис. 3.31) і таким чином, ця сторінка перестала мати зв’язок з мережею.

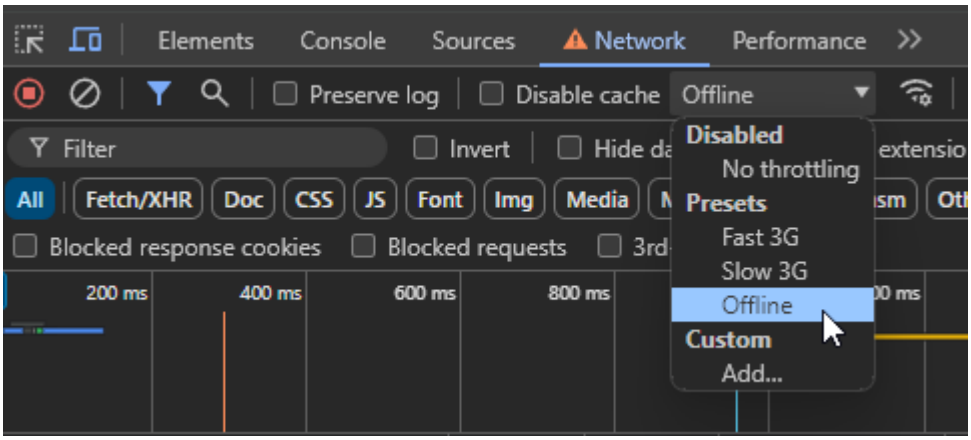

Рисунок 3.31 – Встановлення пресету «Offline» на вкладці Network у Chrome Dev Tools

Після цього було заповнено дані опитування (рис. 3.32).

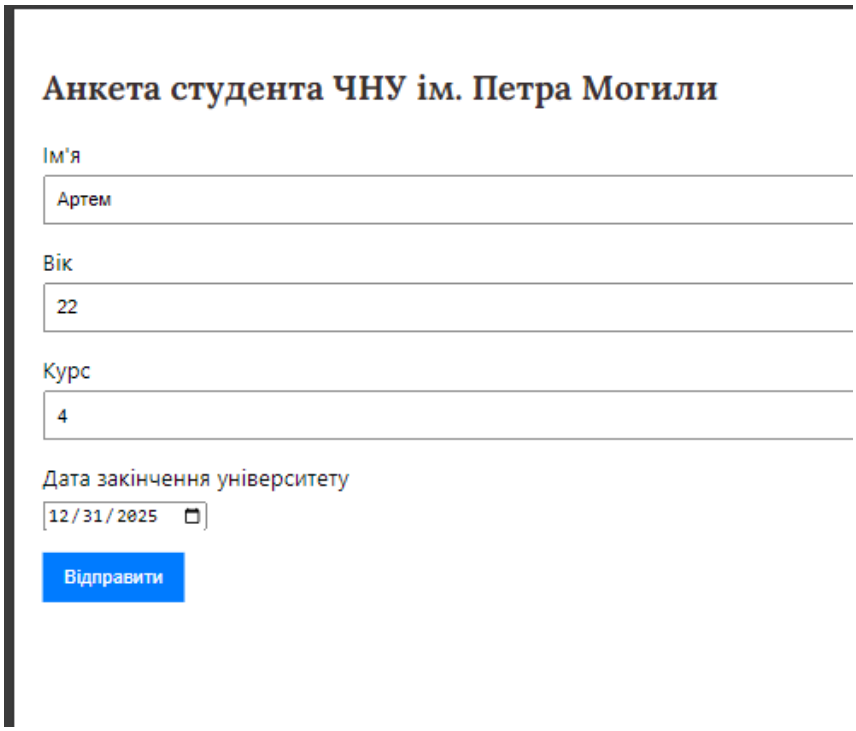

Рисунок 3.32 – Заповнення форми даними

При натисканні кнопки «Відправити», навіть не маючи доступу до інтернету, застосунок зміг зберегти дані до локальної бази даних PouchDB, про що свідчить відповідний спливаючий напис (рис. 3.33).

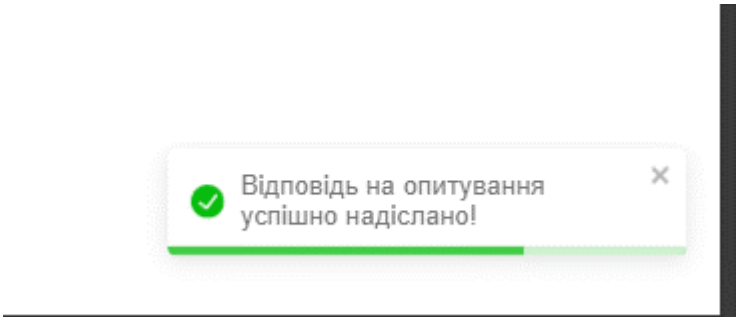

Рисунок 3.33 – Повідомлення про успішне збереження з'явилося

Для того щоб впевнитися, що дані після проходження опитування справді збереглися, було увімкнено доступ до Інтернету та перейдено до Панелі керування.

Після натисканні кнопки «Оновити» можна побачити 1 відповідь на опитування (рис. 3.34).

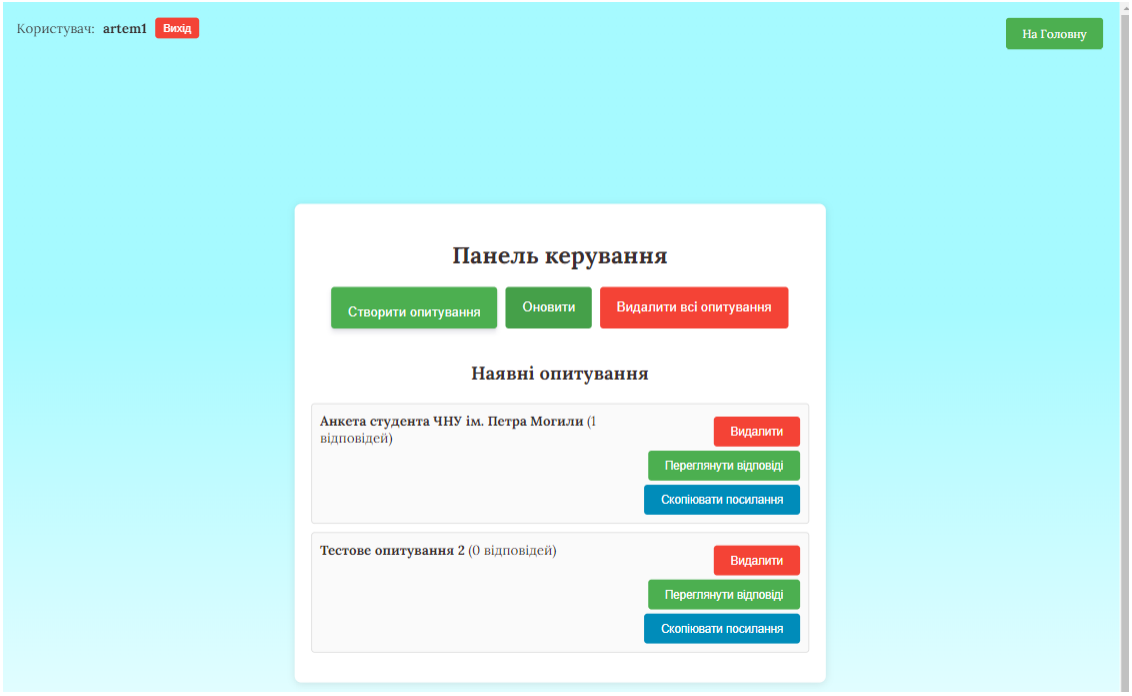

Рисунок 3.34 – Відповідь з’явилася

Для того, щоб впевнитися, що це саме та відповідь, яка була збережена без доступу до інтернету, було натиснуто кнопку «Переглянути відповіді» (рис. 3.35).

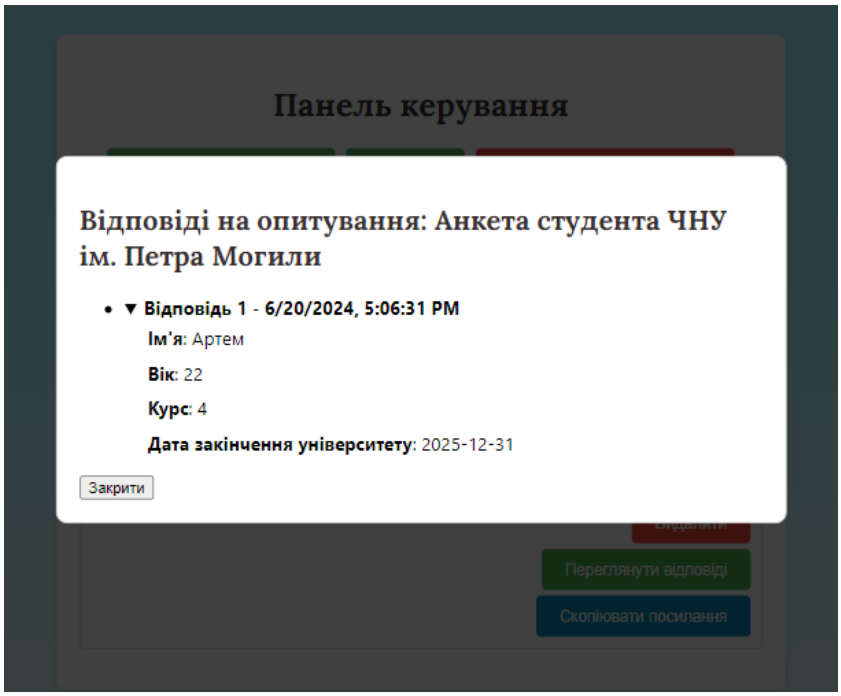

Рисунок 3.35 – Відповідь з’явилася

Після недовгого аналізу даних з відповіді, стало зрозуміло, що це і є та сама

відповідь, яка була збережена до бази даних, не дивлячись на відсутність підключення до мережі Інтернет. Отже, головний функціонал застосунку, через що його і було вирішено розробити, а саме – стабільна робота та можливість проходження опитувань як при стабільному зв’язку, так і без - працює.

Висновки до розділу 3

У третьому розділі було детально розглянуто процес розробки програмного комплексу для анкетування, який здатен функціонувати в офлайн режимі. Цей розділ охоплював усі ключові етапи створення системи, починаючи з проектування архітектури, вибору технологій та закінчуючи реалізацією та тестуванням.

Процес проектування архітектури системи включав визначення основних компонентів, таких як клієнтська частина, серверна частина, база даних та механізми синхронізації даних. Було обрано технології PouchDB для локального зберігання даних на пристроях користувачів та CouchDB для зберігання даних на сервері з метою забезпечення безперебійної роботи системи навіть в умовах відсутнього інтернет-з'єднання.

Клієнтська частина системи була реалізована з використанням бібліотеки React, що забезпечило високу швидкість та інтерактивність інтерфейсу користувача. Використання Redux дозволило ефективно керувати станом додатку, що є критично важливим для спільного редагування анкет різними користувачами. Серверна частина була побудована на основі Node.js та Express, що забезпечило стабільну та масштабовану обробку запитів.

Тестування системи включало як функціональні, так і нефункціональні аспекти. Функціональне тестування підтвердило коректність роботи основних функцій системи, таких як створення, редагування та видалення анкет, а також синхронізація даних між локальною та серверною базами даних. Нефункціональне тестування включало перевірку надійності, продуктивності та безпеки системи. Результати тестування показали, що система здатна ефективно працювати та забезпечувати захист даних користувачів.

Впровадження програмного комплексу може дозволити забезпечити стабільний збір та обробку даних у віддалених регіонах, що є критично важливим для планування та реалізації програм відновлення інфраструктури. Система довела свою ефективність та надійність в умовах обмеженого або відсутнього інтернет-з'єднання, що значно підвищує її практичну цінність.

Таким чином, третій розділ підтвердив, що розроблений програмний комплекс відповідає заявленим вимогам та завданням, забезпечуючи надійний інструмент для збору та обробки даних в умовах кризових ситуацій. Проведене тестування та апробація системи демонструють її готовність до реального використання, що сприяє досягненню основної мети проекту – оперативного та ефективного збору інформації для підтримки процесів відновлення.

ВИСНОВКИ

Розробка програмного комплексу для анкетування, що забезпечує можливість функціювання в офлайн режимі в умовах обмеженого або відсутнього інтернет-з'єднання, є важливим кроком у напрямку підвищення ефективності збору даних у віддалених та постраждалих регіонах. Протягом роботи над цим проектом було досягнуто значних результатів, що дозволяють забезпечити стабільну та надійну роботу системи в різних умовах, включаючи кризові ситуації.

Основним досягненням стало впровадження можливості офлайн-роботи за допомогою технологій PouchDB та CouchDB, що дозволяє зберігати дані на локальних пристроях користувачів та автоматично синхронізувати їх із сервером після відновлення інтернет-з'єднання. Це рішення забезпечує надійність і безпеку даних, зменшуючи ризики їх втрати або пошкодження.

Також важливим аспектом розробки стало забезпечення безпеки даних користувачів. Впровадження механізмів шифрування даних, аутентифікації та авторизації користувачів, а також управління сесіями допомогло створити безпечне середовище для зберігання та обробки конфіденційної інформації. Дотримання правил захисту інформації, визначених законодавством, гарантує високий рівень захисту даних.

Інтерфейс користувача розроблений з урахуванням усіх вимог та принципів інтуїтивної взаємодії, що мінімізує навантаження на очі під час роботи з програмою. Використання приємних кольорів, читабельних шрифтів та інформативного зворотного зв'язку підвищує загальний користувацький досвід.

Технічна безпека та надійність системи були забезпечені завдяки впровадженню резервного копіювання даних, моніторингу продуктивності та використанню сучасних методів захисту від кібератак. Забезпечення безперебійної роботи системи дозволяє гарантувати постійний доступ до даних та зменшує ймовірність збоїв у роботі.

Таким чином, розроблений програмний комплекс для анкетування не лише відповідає сучасним вимогам безпеки та надійності, але й забезпечує комфортні умови для користувачів та працівників, що здійснюють виїзні анкетування. Це сприяє більш ефективному збору та обробці даних, що є критично важливим для планування та реалізації програм відновлення інфраструктури у кризових умовах. Розробка та впровадження цього комплексу є значним внеском у забезпечення своєчасної та точної допомоги постраждалим регіонам.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Структура анкети, типи та види запитань. URL: https://qala-project-1.gitbook.io/vivchennya-potreb-u-navchann-ta-profes-jnomu-rozvi/anketuvannya-1/anketuvannya (дата звернення: 20.04.2024).

2. Basics of KoboToolbox. URL: https://support.kobotoolbox.org/welcome.html (дата звернення: 01.05.2024).

3. Google Forms – Вікіпедія. URL: https://en.wikipedia.org/wiki/Google_Forms (дата звернення: 01.05.2024).

4. Microsoft Forms. URL: https://www.microsoft.com/uk-ua/microsoft-365/online-surveys-polls-quizzes (дата звернення: 04.05.2024).

5. PouchDB та CouchDB (бази даних) – синхронізація даних. URL: https://pouchdb.com/ (дата звернення: 04.05.2024).

6. WebSockets API Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 05.05.2024).

7. Data Versioning – IT Wiki. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/data-versioning (дата звернення: 06.05.2024).

8. Основи UI/UX – DOU. URL: https://dou.ua/forums/topic/42880/ (дата звернення: 05.05.2024).

9. Material Design - документація. URL: https://m3.material.io/get-started (дата звернення: 09.05.2024).

10. OWASP Foundation – безпека застосунка. URL: https://owasp.org/ (дата звернення: 06.05.2024).

11. GDPR – керівництво по безпеці даних для розробників. URL: https://gdpr.eu/ (дата звернення: 09.05.2024).

12. JWT (токен доступу). URL: https://jwt.io (дата звернення: 17.05.2024).

13. Сучасні технології для створення вебзастосунків – JavaScript Weekly. URL: https://javascriptweekly.com/ (дата звернення: 01.05.2024).

14. React (фреймворк) – документація. URL: https://react.dev/learn (дата звернення: 02.05.2024).

15. Redux – управління станом додатку. URL: https://redux.js.org (дата звернення: 15.05.2024).

16. Express.js Documentation. URL: https://expressjs.com (дата звернення: 16.05.2024).

17. Node.js Official Documentation. URL: https://nodejs.org (дата звернення: 18.05.2024).

18. Керівництво з використання DOM. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (дата звернення: 25.05.2024).

19. React Query – управління даними сервера. URL: https://react-query.tanstack.com (дата звернення: 14.05.2024).

20. Basic Tutorial for PouchDB and CouchDB – usePouchDB. URL: https://terreii.github.io/use-pouchdb/docs/introduction/pouchdb_couchdb (дата звернення: 10.05.2024).

21. Introduction to PouchDB. URL: https://pouchdb.com/guides/ (дата звернення: 13.05.2024).

22. Data Replication: Tools and Techniques for Managing Distributed Information. URL: https://www.amazon.com/Data-Replication-Techniques-Distributed-Information/dp/0471157546

23. Designing User Interfaces: Exploring User Interfaces, UI Elements, Design Prototypes and the Figma UI Design Tool (English Edition). URL: https://www.amazon.com/Designing-User-Interfaces-Exploring-Prototypes-ebook/dp/B09B3HHS3D

24. An introduction to the npm package manager. ULR: https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager#an-introduction-to-the-npm-package-manager (дата звернення: 19.05.2024).

25. DALL-E (нейронна мережа) – Вікіпедія. URL: https://uk.wikipedia.org/wiki/DALL-E (дата звернення: 24.05.2024).

ДОДАТОК А
СТОРІНКА HOME

```
const getRandomPosition = (minX, maxX, minY, maxY) => {
  const x = Math.random() * (maxX - minX) + minX;
  const y = Math.random() * (maxY - minY) + minY;
  return { x, y };
};

function Home() {
  useEffect(() => {
    const logos = document.querySelectorAll(".logo-ngo");
    const minX = window.innerWidth * -0.2;
    const maxX = window.innerWidth * 0.12;
    const minY = window.innerHeight * 0;
    const maxY = window.innerHeight * 0.2;

    logos.forEach((logo) => {
      const { x, y } = getRandomPosition(minX, maxX, minY, maxY);
      logo.style.transform = `translate(${x}px, ${y}px)`;

      anime({
        targets: logo,
        translateX: x + Math.random() * 20 - 10,
        translateY: y + Math.random() * 20 - 10,
        duration: 10000,
        direction: "alternate",
        loop: true,
        easing: "easeInOutQuad",
      });
    });

    anime({
      targets: ".logo-ngo",
      opacity: [0, 0.6],
      duration: 2000,
      easing: "easeInOutQuad",
    });
  }, []);

  return (
    <div className="main">
      <div className="landing-view">
        <div className="landing-view-content">
          <div className="header">
            <div className="logo">
              <img src="./logo3.png" alt="Logo" />
            </div>
```

```
            <div className="buttons">
              <Link to="/about">
                <Button className="button--large button-margin">Про
нас</Button>
              </Link>
              <Link to="/contactus">
                <Button className="button--large button-margin">
                  Зв'язатися з нами
                </Button>
              </Link>
              <Link to="/login">
                <Button className="button--large button-
margin">Увійти</Button>
              </Link>
              <Link to="/register">
                <Button className="button--large button-margin">
                  Створити аккаунт
                </Button>
              </Link>
            </div>
          </div>
          <h3>Інструмент №1 для збору даних</h3>
          <h1>DATA GATHER</h1>
          <h3>
            Приєднуйтесь! Нам вже довіряють десятки гуманітарних організацій
          </h3>

          <div className="ngo-logos">
            {/* <img src="./medWBorders.png" alt="medWBorders"
className="logo-ngo" /> */}
            <img src="./icrc.png" alt="ICRC" className="logo-ngo" />
            <img src="./unicef.png" alt="Unicef" className="logo-ngo" />
            <img src="./ocha.png" alt="OCHA" className="logo-ngo" />
            <img src="./caritas.png" alt="caritas" className="logo-ngo" />
            <img src="./mriyDiy.png" alt="mriyDiy" className="logo-ngo" />
            <img src="./sds.png" alt="SDS" className="logo-ngo" />
            <img src="./smile.png" alt="smile" className="logo-ngo" />
            <img src="./shedryk.png" alt="shedryk" className="logo-ngo" />
          </div>
```

ДОДАТОК Б
ФУНКЦІЯ ОНОВЛЕННЯ СПИСКУ ОПИТУВАНЬ

```
 PouchDB.plugin(PouchDBFind);

function Dashboard() {
  const [surveys, setSurveys] = useState([]);
  const [selectedSurvey, setSelectedSurvey] = useState(null);
  const { user, logout } = useAuth();
  const navigate = useNavigate();

  const db = useMemo(() => new PouchDB('survey_db'), []);

  const fetchSurveys = async () => {
    try {
      await db.info();
      const result = await db.find({
        selector: { username: user.username }
      });
      const surveysWithResponsesCount = await
Promise.all(result.docs.map(async (survey) => {
        const responses = await db.find({
          selector: { surveyId: survey._id }
        });
        return { ...survey, responseCount: responses.docs.length, responses:
responses.docs };
      }));
      setSurveys(surveysWithResponsesCount);
    } catch (err) {
      console.error(err);
    }
  };

  useEffect(() => {
    fetchSurveys();
  }, [user.username, db]);
```

ДОДАТОК В
ФУНКЦІЯ ВИДАЛЕННЯ ВСІХ ОПИТУВАНЬ

```
const handleDeleteAll = async () => {
   try {
     await db.info();
     const result = await db.find({
       selector: { username: user.username }
     });
     const docs = result.docs.map(doc => ({
       _id: doc._id,
       _rev: doc._rev,
       _deleted: true
     }));
     await db.bulkDocs(docs);
     setSurveys([]);
     toast.success('All surveys deleted successfully!', {
       position: "bottom-right",
       autoClose: 5000,
       hideProgressBar: false,
       closeOnClick: true,
       pauseOnHover: true,
       draggable: true,
       progress: undefined,
       theme: "light",
     });
   } catch (err) {
     toast.error('Error deleting surveys', {
       position: "bottom-right",
       autoClose: 5000,
       hideProgressBar: false,
       closeOnClick: true,
       pauseOnHover: true,
       draggable: true,
       progress: undefined,
       theme: "colored",
     });
     console.error(err);
   }
 };
```