МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток для проведення маркетингових досліджень»

за спеціальністю 122 «Комп'ютерні науки та інформаційні технології», освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТз-51с Бутурлім В'ячеслав Борисович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ « ___»_____ 2020 р.

Науковий керівник _____ к.т.н. Алексенко О.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ _____
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Бутурлім В'ячеслав Борисович

1 Тема роботи *Web-додаток для проведення маркетингових досліджень*

керівник роботи *Алексенко Ольга Василівна, к.т.н.*,

затверджені наказом по університету від « 15 » травня 2020 р. № 0582-III

2 Строк подання студентом роботи «6» червня 2020 р.

3 Вхідні дані до роботи *технічні та методичні матеріали*

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) реферат, вступ, аналіз предметної області, постановка задачі та методи дослідження, моделювання, розробка проекту, тестування web-додатку для проведення маркетингових досліджень, висновок, список використаних джерел, додатки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Мета, задачі, актуальність, аналіз предметної області, функціональні вимоги, постановка задачі, засоби реалізації, діаграми IDF0, діалогами варіантів використання, розробка макету, практична реалізація, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____
_

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз предметної області	До 07.02.2020	
2.	Проектування засобів для реалізації системи	До 27.03.2020	
3.	Реалізація компонентів інформаційної системи	До 13.04.2020	
4.	Впровадження в експлуатацію	До 08.05.2020	
5.	Здача пояснювальної записки та файлів розробленого проекту	До 06.06.2020	

Студент _____ Бутурлім В.Б.
(підпис)

Керівник роботи _____ к.т.н. Алексенко О.В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра: Web-додаток для проведення маркетингових досліджень

Кваліфікаційна робота бакалавра присвячена розробці web-додатку для проведення маркетингових досліджень. Було проведено дослідження та проведений аналіз всіх головних аспектів інформаційних потоків.

Пояснювальна записка містить: 5 - розділів, вступ, висновок, 3 додатки та список використаних джерел, включає 87 сторінок, 4 таблиці, 51 рисунок, 17 – джерел.

В першому розділі наведено аналіз переметної області, аналіз додатків аналогів, та актуальність проблеми дослідження.

В другому розділі здійснюється вибір засобів реалізації написання web-додатку, мета та задачі створення web-додатку.

Третій розділ представлено структурно-функціональний аналіз та моделювання варіантів використання для web-додатку.

Четвертий розділ пояснювальної записки описує процес розробки web-додатку.

В результаті проведеної роботи розроблений web-додаток для проведення маркетингових досліджень, що надасть змогу потенційним клієнтам проводити маркетингові дослідження методом опитування у зручній формі.

Ключові слова: опитування, web-додаток, маркетингові дослідження.

ЗМІСТ

ВСТУП

У сучасному світі ні одна з компаній не зможе існувати в мережі ринкових відносин без проведення маркетингових досліджень. Потреби людей завжди великі, а ресурси якими володіють підприємці не завжди можуть задовольнити всіх. На допомогу прийшли маркетингові дослідження я одна з головних функцій керування інформацією в умовах ринку.

За допомоги маркетингових досліджень підприємці мають змоги відстежувати потреби покупців, як правило маркетингові дослідження проводяться коли: підприємство не досягає запланованих цілей; поступається конкуренту; формується новий бізнес план.

Часто основна мета маркетингових досліджень надати характеристику ринку, процесів та явищ. Найчастіше маркетингові дослідження поділяються на аналіз ринку, аналіз товарів, конкурентів, середовищ маркетингу, ціни на товар чи послуги та якість обслуговування.

Метою дипломного проекту є створення web-додатка із зручним інтерфейсом для проведення маркетингових досліджень методом опитування, який дасть змогу кожному підприємцю проводити потрібні для нього маркетингові дослідження.

Під час створення web-додатку проведення маркетингових досліджень, для реалізації мети потрібно вирішити наступні задачі:

- провести аналіз аналогів web-додатку для визначення яким характеристикам має відповідати розроблюваний продукт;
- провести аналіз предметної області для вибору технологій та інструментів реалізації;
- розробити план робіт з розробки web-додатку та оцінити можливі ризики;
- спроектувати web-додаток, реалізувати та провести його тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

В сучасному світі маркетингові дослідження [14] мають велике значення. Вони надають можливість дізнатись про актуальність проблем, уподобання товарів, якість обслуговування. Допомагають визначити напрямок розвитку ринку.

Більшість компаній-гігантів використовують маркетингові дослідження для аналізу ринку, але мало невеликих компаній чи магазинів користуються маркетинговими дослідженнями. Можливість проводити невеликі маркетингові дослідження надасть змогу підприємствам малого та середнього бізнесу збільшити свій прибуток на основі даних із аналізу ринку.

Висока актуальність проблем, недостатньо висвітлення питань вивчення та проведення маркетингових досліджень за допомогою автоматизованих систем підштовхують то створення web-додатків для проведення маркетингових досліджень.

Інформаційні потоки маркетингових досліджень поділяються на: визначення ринкових цілей підприємства, визначення факторів підприємства, визначення стратегій, визначення стану комплексу маркетингу, положення на ринку, визначення поведінки споживачів.

В сучасному світі маркетингові дослідження мають велике значення. Вони надають можливість дізнатись про актуальність проблем, уподобання товарів, якість обслуговування. Допомагають визначити напрямок розвитку ринку.

Більшість компаній-гігантів використовують маркетингові дослідження для аналізу ринку, але мало невеликих компаній чи магазинів користуються маркетинговими дослідженнями. Можливість проводити невеликі маркетингові дослідження надасть змогу невеликим компаніям збільшити свій прибуток, аналізуючи ринок.

Запропонований Web-додаток надасть змогу проводити дослідження любої аудиторії людей, що дозволить всім проводити опитування про свій товар чи якість обслуговування.

Очікувані переваги використання додатку проведення маркетингових досліджень:

- можливість бачити зацікавлена аудиторія чи ні;
- направлення на цільову аудиторію;
- можливість проведення повторних та регулярних опитувань.

1.2 Аналіз сайтів проведення маркетингових досліджень методом опитування

Процес розробки web-додатку неприйнятний без попередньої постановки завдань та цілей. Потрібно провести ретельний аналіз web-додатків аналогів, для виявлення слабких та сильних сторін проекту.

Для створення додатку було проаналізовано додатки аналоги:

- web-додаток Digdata [15] ресурс https://digdata.com.ua/ (рис. 1.1);
- web-додаток Socis [16] ресурс http://socis.kiev.ua/ (рис. 1.2);
- web-додаток Google forms [17] ресурс https://www.google.com/intl/uk_ua/forms/about/ (рис. 1.3).

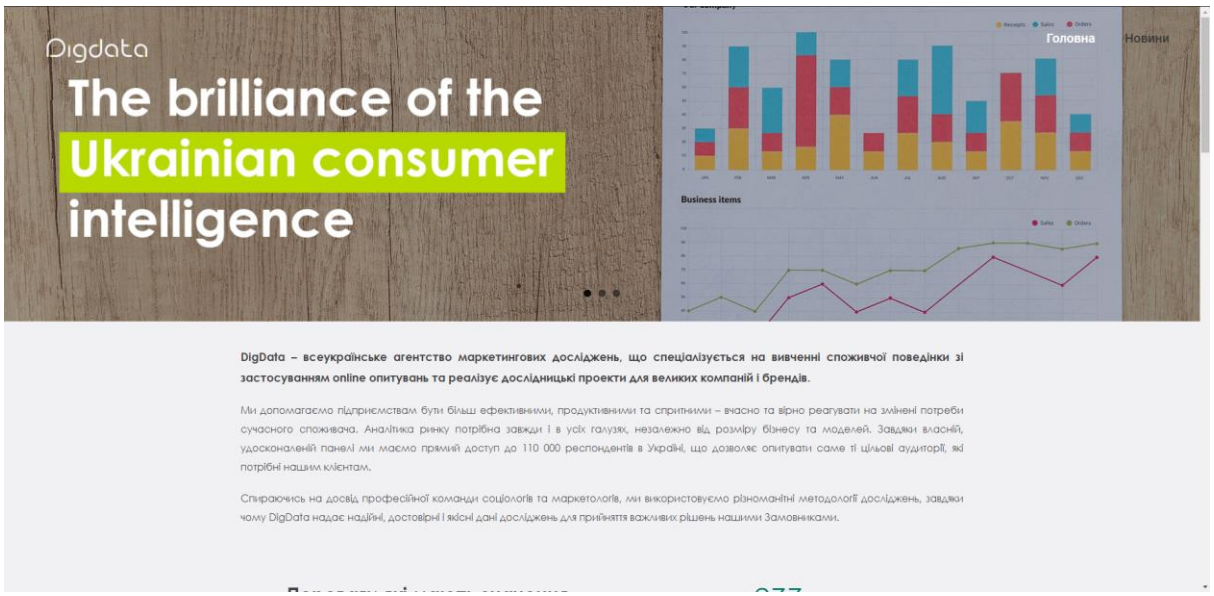


Рисунок 1.1 – *сторінка web-додатку Digdata*

Рисунок 1.2 – *сторінка web-додатку Socis*

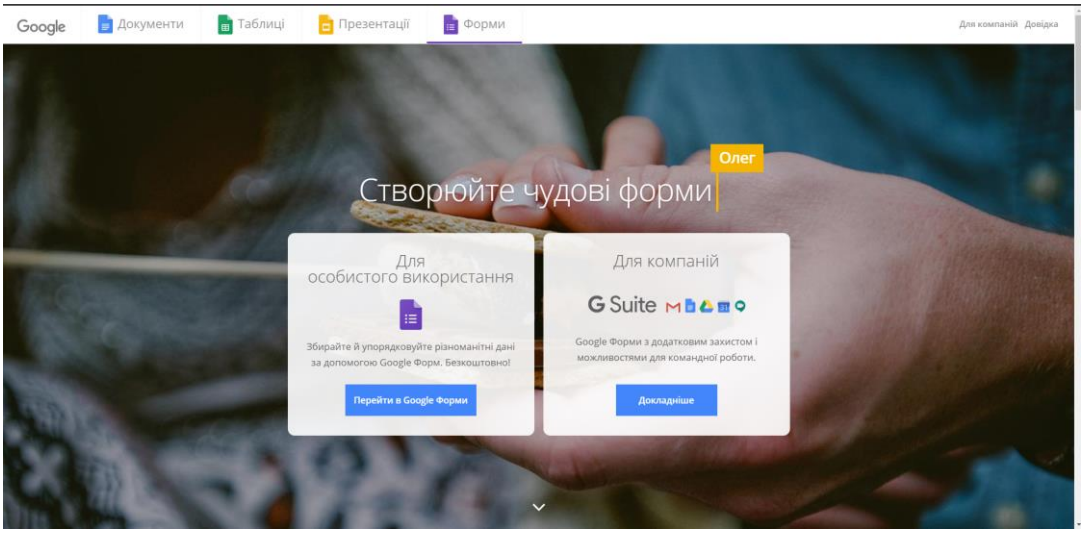


Рисунок 1.3 – *сторінка web-додатку Google forms*

Проаналізувавши аналогі web-додатків проведення маркетингових досліджень було створено порівняльний аналіз (табл.1.1), під час якого виявлено основні позитивні та негативні сторони, що допоможе уникати даних помилок під час створення дипломного проекту.

Недоліки:

– недопрацьований контент;

– відсутність можливості створення опитувань;

– відсутня можливість проведення опитувань конкретної аудиторії;

До основних переваг аналогічних web-додатків можна віднести:

– зручність навігації;

– добра кольорова палітра web-додатків;

– компактність всіх елементів web-додатків;

– відсутність непотрібної реклами;

– можливість безкоштовного проведення опитувань.

Таблиця 1.1 – Аналіз розглянутих комерційних сайтів фотографів

Критерії	web-додаток Digdata	web- додаток Socis	web-додаток Google forms
Відсутність непотрібної реклами	+	+	+

Продовження таблиці 1.1 – Аналіз розглянутих комерційних сайтів фотографів

Вдалий підбір кольорової палітри	-	+	+
Можливість безкоштовного проведення опитувань	-	-	+
Можливість проведення опитування окремих груп людей	-	-	+
Наявність мовних версій	-	-	+
Зручна навігація	+	-	+
Зручний інтерфейс створення опитувань	-	-	+
Компактність елементів контексту	+	+	+

2 ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУ

2.1 Мета та задачі

Головною метою роботи є створення web-додатку для проведення маркетингових досліджень, що дасть змогу проводити опитування різних напрямків.

Web-додаток повинен бути зручний у використанні. Надавати змогу бачити результати проведення опитувань, та кількість людей яка не була зацікавлена в опитуванні чи проігнорувала його.

Для виконання даної задачі необхідно виконати наступні задачі:

- провести аналіз аналогів сайтів опитування та визначити їх переваги і недоліки;
- визначити програмні та технічні засоби для реалізації;
- спроектувати структуру web-додатку, користувацький інтерфейс та його базу даних;
- розробити web-додаток маркетингових досліджень та провести тестування його роботи.

2.2 Вибір засобів реалізації

Web-розробка – основний процес створення web-сайтів та web-додатків. Головними етапами процесу є web-дизайн, верстка сторінок, програмування back-end та front-end, а також настройка конфігурацій web-сервера.[1]

Розробка web-додатку ділиться на декілька етапів:

- планування та проектування web-додатку;
- розробка структури web-додатку [2], вибір мови програмування та програмних засобів для створення web-додатку, вибір хостингу;
-

- розробка web-додатку, створення основних сторінок та написання коду web-додатку;
- падлючення основних API додатків;
- тестування web-додатку;
- підтримка web-додатку, обслуговування та підтримка програмної мови.

В результаті аналізу сучасних засобів розроблення було обрано такі основні програмні засоби для реалізації web-додатку проведення маркетингових досліджень:

SendGrid - сервіс для відправлення електродних повідомлень, та відстеження їх стану, для зручного написання web-додатків в яких потрібно відправляти електронні повідомлення.[6]

Stripedocs - сервіс для реалізації оплати банківською картою, з можливістю підключення та використання в web-додатах.[5]

Google API - сервіс для реалізації авторизації з допомогою акаунту Google, для більш зручного використання web-додатків.[4]

JavaScript - мультіпарадігменна мова програмування. Вона підтримує об’єктно-орієнтований, імперативний та функціональний стиль.

JavaScript часто застосовується як вбудована мова програмування для створення доступу до об'єктів web-додатків. Часто використовується в браузерах як мова сценаріїв, та для додавання web-сторінкам інтерактивності.

Характерні риси які присутні мові програмування JavaScript:

- слаба типізація;
- динамічна типізація;
- автоматичне керування пам’яттю.

На мову програмування JavaScript мали вплив дуже багато інших мов, під час розробки була мета зробити просту для зрозуміння мову програмування схожу на Java. Мовою програмування не володіють ні компанії чи організації, що робить її унікальною в порівнянні з другими мовами програмування.[13]

Node.js – програмна платформа з відкритим кодом (реалізую перетворення JavaScript в машинний код), тим самим JavaScript перетворюється в мову загального призначення. Дає змогу використовувати пристрої введення та виведення даних через API, та підключати різні бібліотеки, написані на різхних мовах програмування. Переважно Node.js виконую роль web-серверу. Основою Node.js є асинхронне програмування.

React – бібліотека мови програмування JavaScript з відкритим кодом для розробки інтерфейсів користувачів. Часто використовується для розробки мобільних додатків. Основна мета – висока швидкість та простота. Бібліотека React розробляється та підтримується Facebook, Instagram. [12]

MongoDB – система керування базами даних, має відкритий код, та не потребує опису схем таблиць. Код MongoDB реалізований на мові програмування C++. Підтримує збереження документів в JSON, гнучка мова для формування запитів, забезпечує ефективне зберігання великих бінарних об’єктів.

3 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

Після проведення аналізу предметної області, було визначено необхідні дані для реалізації web-додатку для проведення маркетингових досліджень. Були обрані методи реалізації, розглянуті аналоги інших додатків та визначені їх недоліки та переваги. Отже наступним етапом є моделювання web-додатку.

3.1 Структурно-функціональне моделювання web-додатку

Першим кроком процесу моделювання web-додатку для проведення маркетингових досліджень, повинна бути строрення контекстної діаграми A-0 процесу розробки web-додатку для проведення маркетингових досліджень. Діаграма A-0 показує взаємини між основними типами даних, проводячи їх розмежування. Таким чином, діаграма A-0 являє собою загальний вигляд системи, що вивчається. В процесі була розроблена діаграма (рис. 3.1), це діаграма верхнього рівня.

Діаграма A-0 складається лише з одного головного блоку, вхідних та відхідних даних та була розроблена в ERwin Process Modeler.

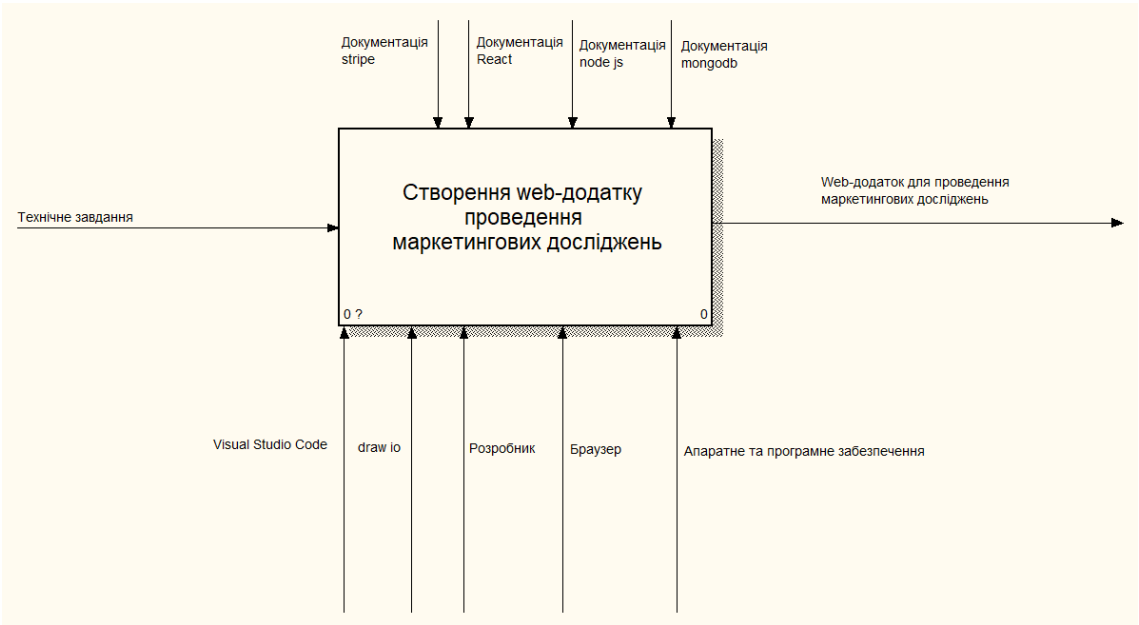


Рисунок 3.1 – *Контекстна діаграма процесу створення web-додатку*

Після опису процесу, була виконана його декомпозиція, де процес створення веб-додатку розбитий на 3 основні частини:

- дослідження предметної області;
- практична реалізація;
- тестування та підтримка продукту.

При дослідженні предметної облізлі ми маємо такі дані:

- вхідні дані: технічне завдання,
- вихідні дані: Web-додаток для проведення маркетингових досліджень.

За контроль відповідають:

- документація React;
- документація node js;
- документація stripe;
- документація mongodb.

Інструменти:

- draw io – для розробки макету;
- Visual Studio Code – для написання основного коду web-додатку;
- Браузер підключення API Google, SendGrid, stripe;
- Розробник – для проведення аналізу існуючих додаткі, дослідження предметної області та написання основного коду;
- апаратне та програмне забезпечення – для підтримки роботи web-додатку.

Діаграма декомпозиції першого рівня представлена на рис.3.2.

Першим етапом декомпозиції контекстної діаграми представляє собою розбиття кожного етапу на перелік робіт, результатом яких є вхідні данні для наступного етапу.

Проведення дослідження предметної області охоплює в себе аналіз та пошук аналогічних web-додатків в мережі Інтернет, після кого ми отримаємо характеристику аналогів.

Проведемо аналіз переваг та недоліків аналогічних web-додатків після чого сформуємо вимоги до нашого web-додатку. В кінці цього етапу розробимо макет web-додатку за допомоги інструменту draw io.

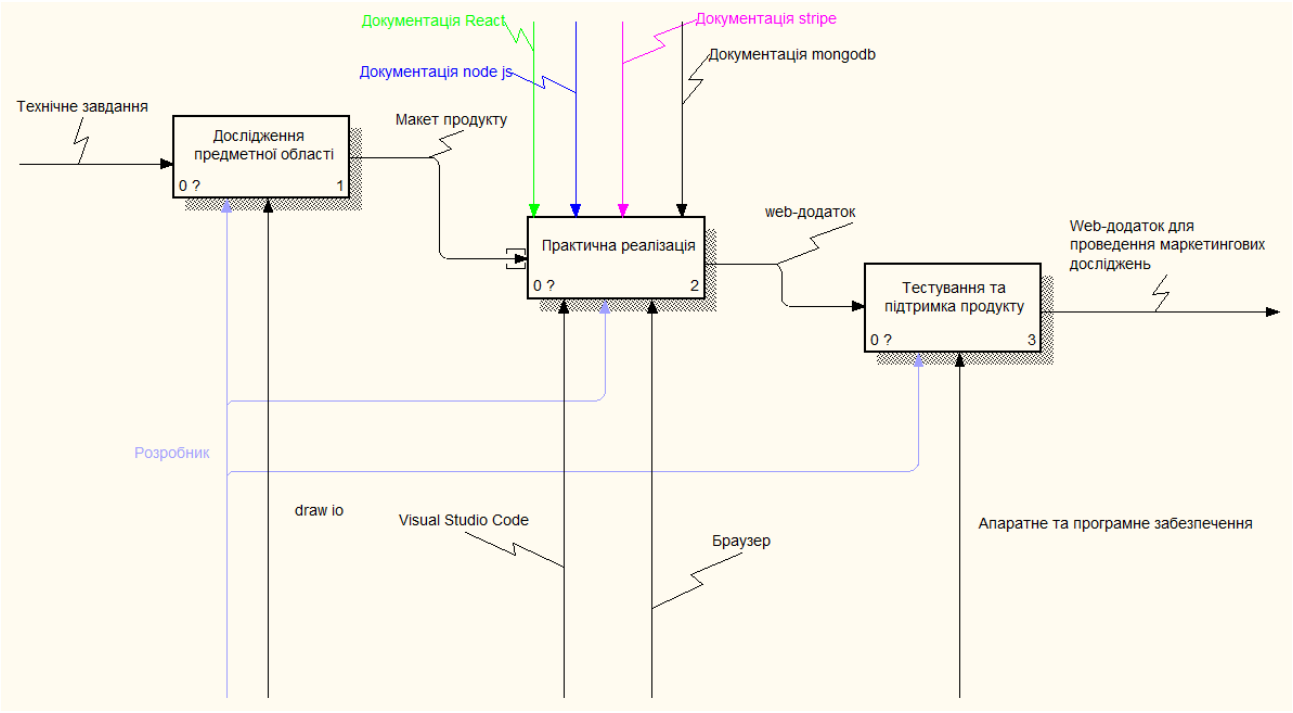


Рисунок 3.2 – *Декомпозиція процесу створення web-додатку*

Декомпозиція під процесу «Дослідження предметної області» представлена на рис.3.3.

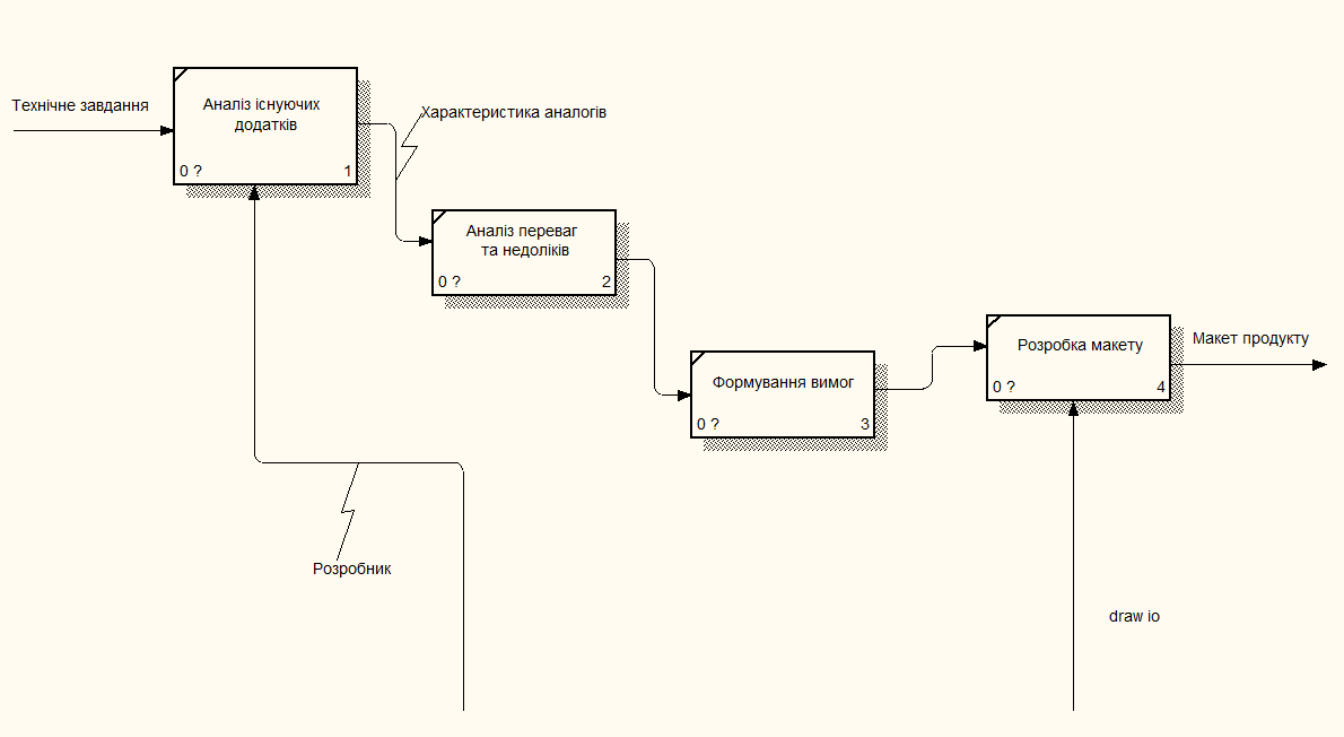


Рисунок 3.3 – *Декомпозиція під процесу «Дослідження предметної області»*

Проведення дослідження предметної області маємо такі вхідні та вихідні дані:

- вхідні дані: технічне завдання web-додатку,
- вихідні дані: макет продукту web-додатку.

Інструменти:

- розробник
- draw io.

Другим етапом декомпозиції практичної реалізації, яка включає в себе створення дизайну на основі отриманого макету, підключення API: Google, SendGrid, stripe, та написання основного коду web-додатку.

Діаграма декомпозиції підпроцесу «Практична реалізація» представлена на рис.3.4.

- вхідні дані: макет продукту,
- вихідні дані: web-додаток.

За контроль відповідають:

- документація React;
- документація node js;
- документація stripe;
- документація mongodb;
- документація material-ui.

Інструменти:

- Visual Studio Code;
- браузер;
- засоби розробки.

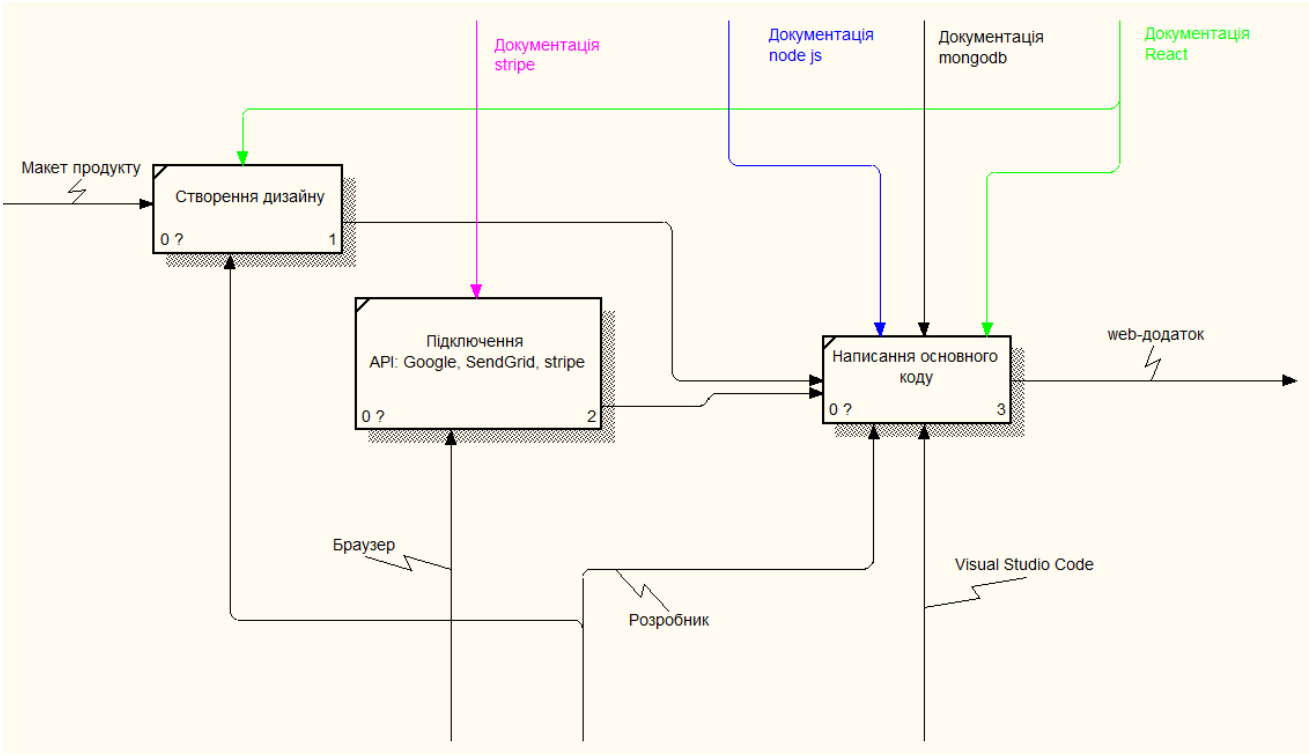


Рисунок 3.4 – *Декомпозиція під процесу «Практична реалізація»*

Третім етапом декомпозиції тестування та підтримка програмного продукту, які включає в себе тестування web-додатку, виправлення помилок, розробку програмної документації та підтримку продукту.

Діаграма декомпозиції процесу «Тестування та підтримка продукту» представлена на рис.3.5.

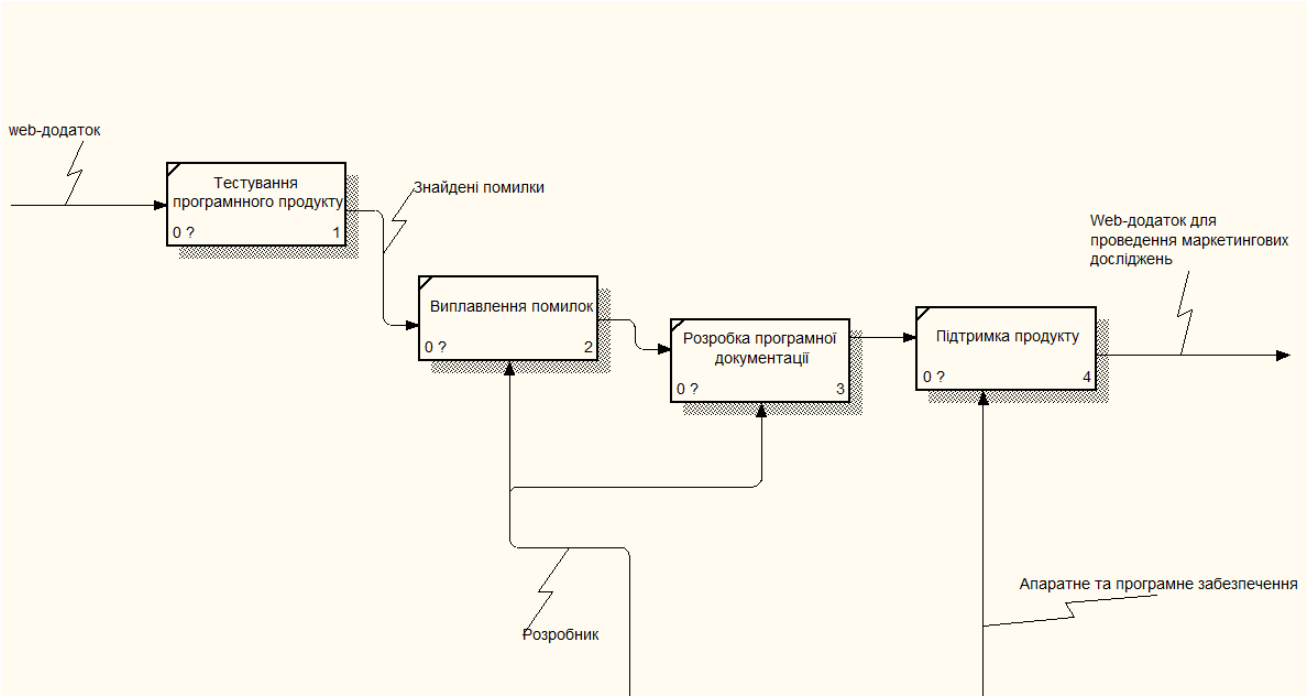


Рисунок 3.5 – *Декомпозиція процесу «Тестування та підтримка продукту»*

- вхідні дані: web-додаток,
- вихідні дані: Web-додаток для проведення маркетингових досліджень.

Інструменти:

- апаратне та програмне забезпечення.

3.2 Моделювання варіантів використання web-додатку

Для побудови діаграми було використано двох «акторів»: «Гість», «Цільова аудиторія».

До основних операцій належить:

- авторизація;
- створення опитувань;
- видалення опитування;
- переглад опитувань;

- надсилання опитувань;
- надання відповідей на опитування.

Автор «Гість» після авторизації в додатку має змогу створювати опитування або видаляти, надсилати їх, після чого отримувати результати опитувань та переглядати їх. Має повний доступ до функціоналу web-додатку після авторизації.

Автор «Цільова аудиторія» це велика кількість людей яким надсилаються опитування і вони мають змогу тільки надавати відповіді на опитування, не мають доступу до функціоналу додатку.

Діаграма варіантів використання [11] була розроблена з використанням інструменту draw io.

Варіанти використання:

- авторизація – необхідна для збереження даних, та для доступу до повного функціоналу web-додатку;
- створення опитувань – створення простих опитувань цільової аудиторії;
- видалення опитування – після проведення опитування та втрати актуальності опитування є можливість видалити опитування;
- перегляд опитувань – погляд створених опитувань та їх результатів;
- надсилання опитувань – надсилання опутувань цільовій аудиторії;
- надання відповідей на опитування – проходження опитувань.

На основі даних була розроблена та описана діаграма варіантів використання web-додатку для проведення маркетингових досліджень представлена на рисунку 3.6.

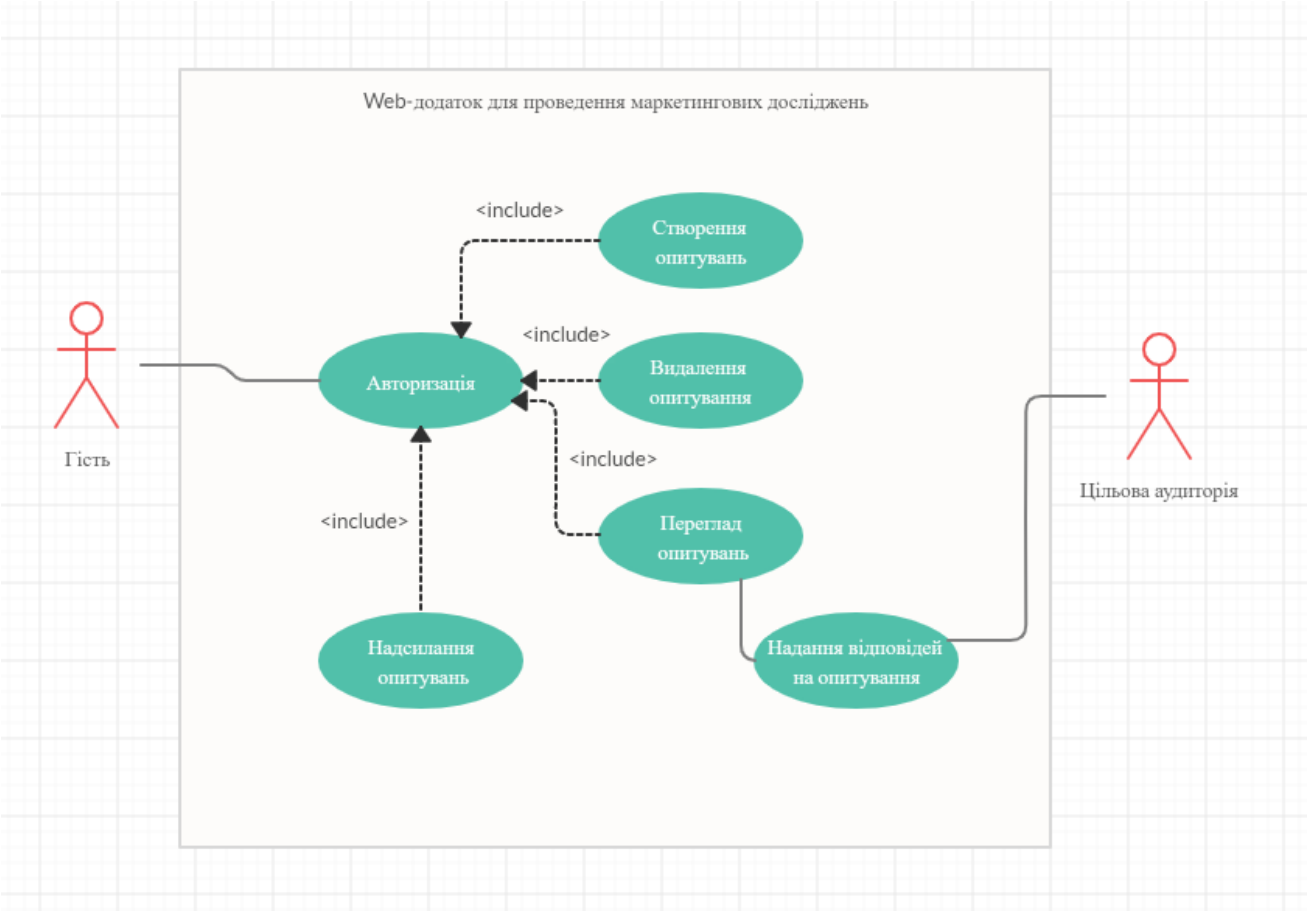


Рисунок 3.6 – *Діаграма варіантів використання web-додатку*

4 РОЗРОБКА WEB-ДОДАТКУ

4.1 Встановлення основних копононетів

Першим етапом розробки web- є встановлення Visual Studio Code для зручного написання коду програми.

Visual Studio Code — программа для зручного написання та редагування коду, підтримує всі мови програмуванням для створення web-додатків, є безкоштовною та зручною у використанні, що надає можливість розробляти web-додатки та працює на всіх операційних системах.

Розроблена кома пінією Microsoft весною 2015, перше крос-пладформне середовище розробки web-додатків.[10]

Основний сайт з документацією для програми https://code.visualstudio.com/docs зображений на (рис 4.1).

Після завантаження програми встановлюємо (рис 4.2) її та проводимо основні налаштування про роботи з React та Node Js.

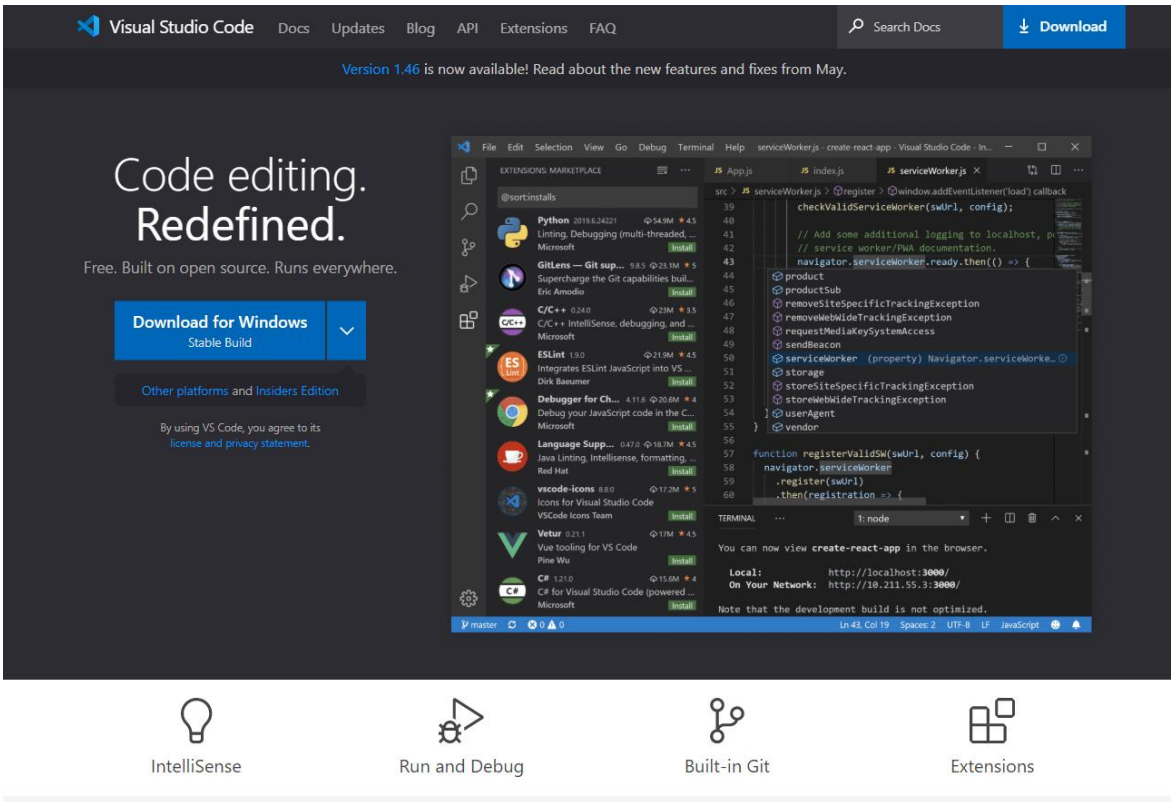


Рисунок 4.1 – *сайт Visual Studio Code*

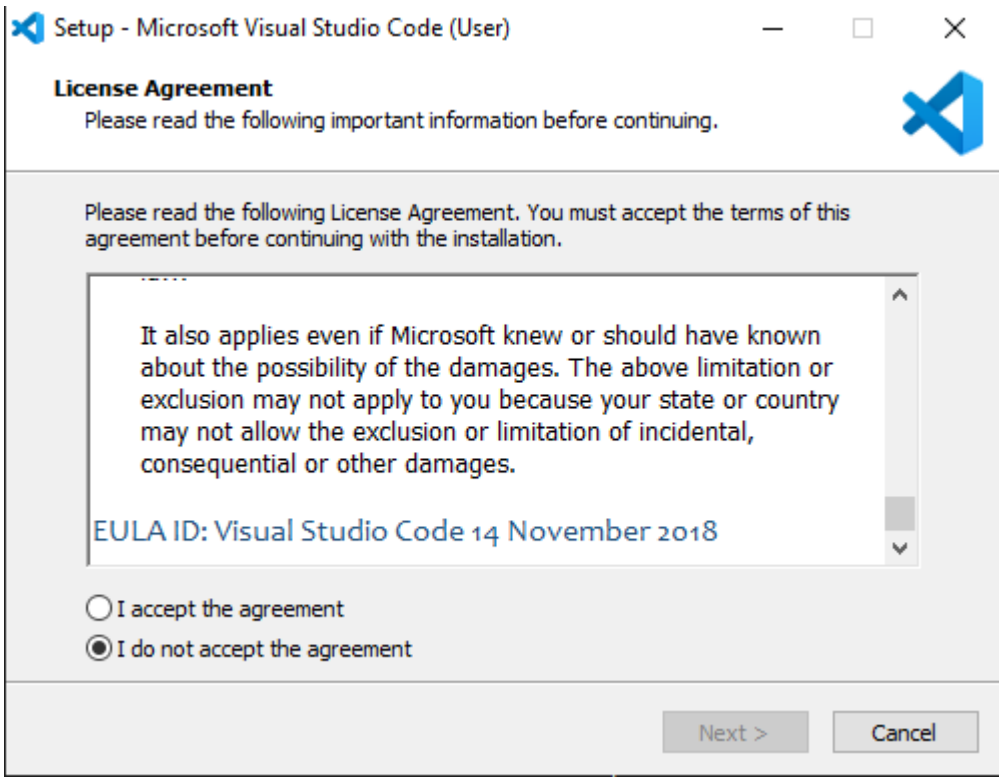


Рисунок 4.2 – *встановлення Visual Studio Code*

Наступним кроком будемо прямцювати з Node Js.

Node.js- асинхронне JavaScript оточення, створене для програмування web-додатків для обробляння багатьох одночасних з'єднань.[9]

Завантажуємо node js (рис. 4.3) після чого встановлюємо (рис. 4.4).

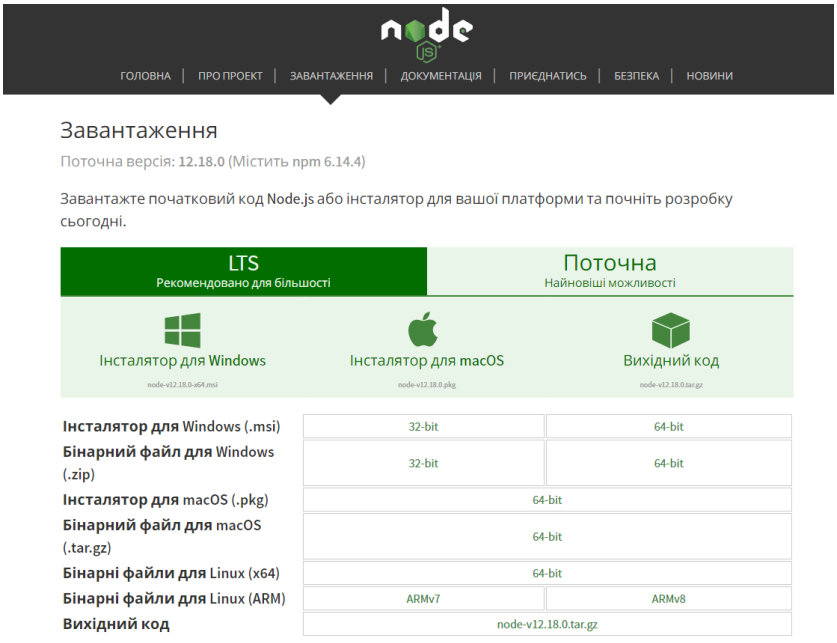


Рисунок 4.3 – *сайт node js*

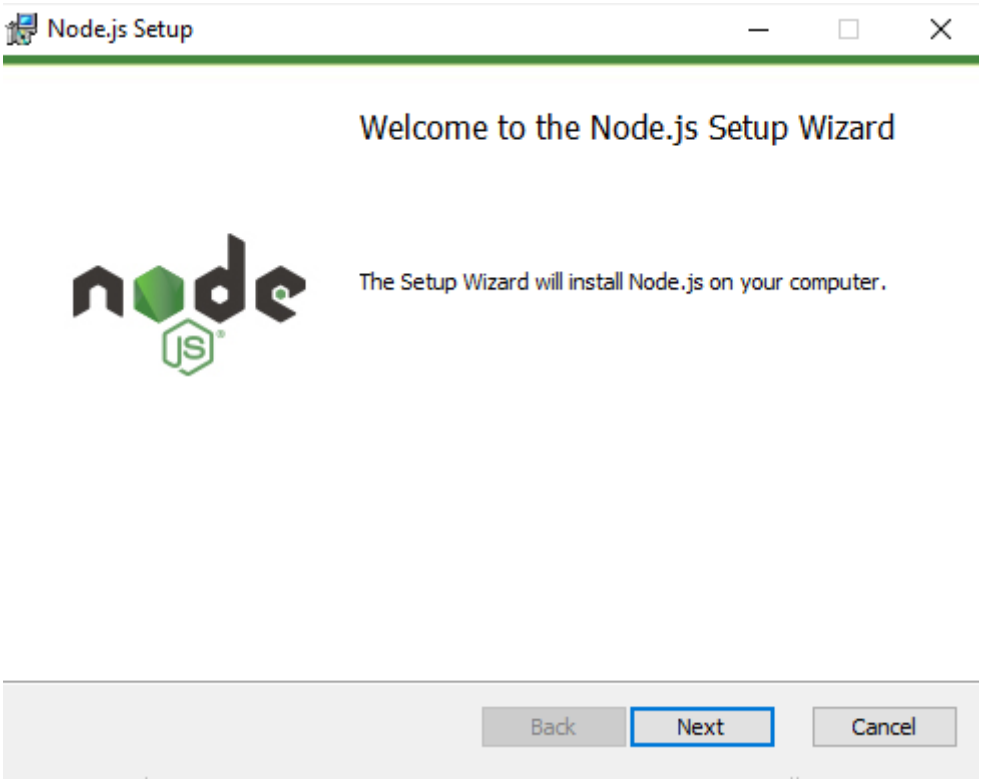


Рисунок 4.4 – *встановлення node js*

4.2 Встановлення бібліотек та створення web-додатку

Наступним кроком потрібно встановити React бібліотеку для створення програм. Для цього відкриваємо програму Visual Studio Code переходимо до консолі та прописуємо npm install -g create-react-app (рис.4.5).

Тепер переходимо до створення React програми в командній строчці прописуємо create-react-app client [8] (рис.4.6).

Після встановлення додатку запускаємо його командою npm start, результатом успішного встановлення бібліотек та створення додатку React бачимо у вікні браузеру (рис 4.7)

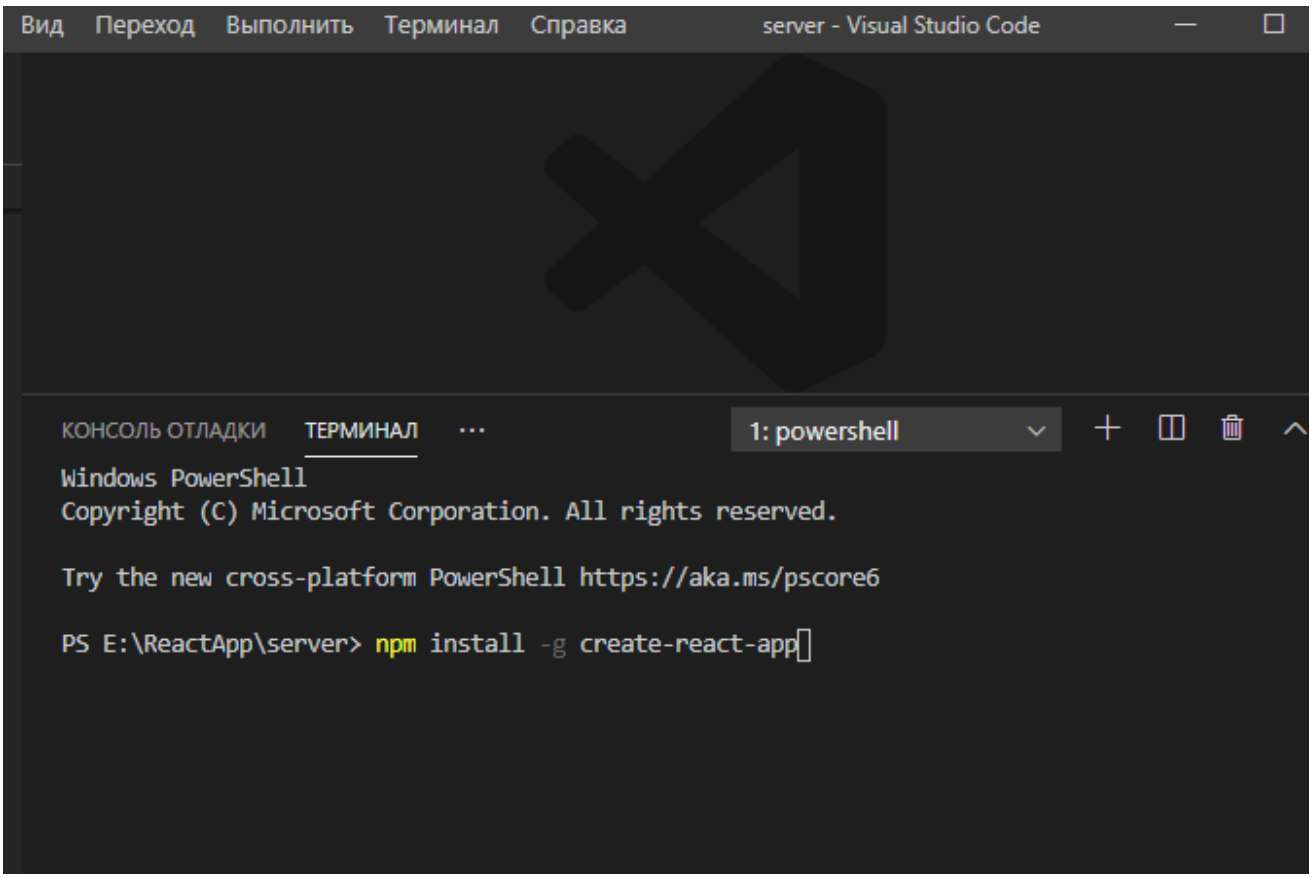


Рисунок 4.5 – *встановлення React бібліотеки*

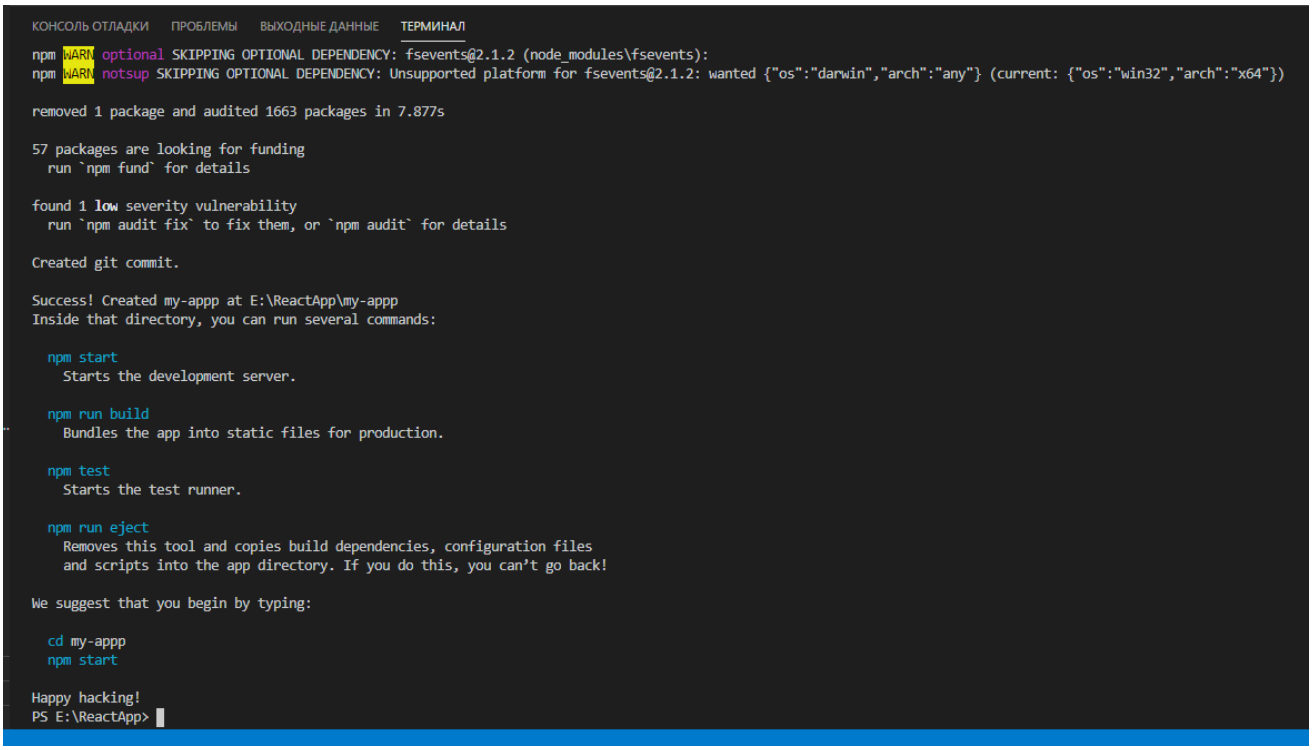


Рисунок 4.6 – *встановлення React додатку*

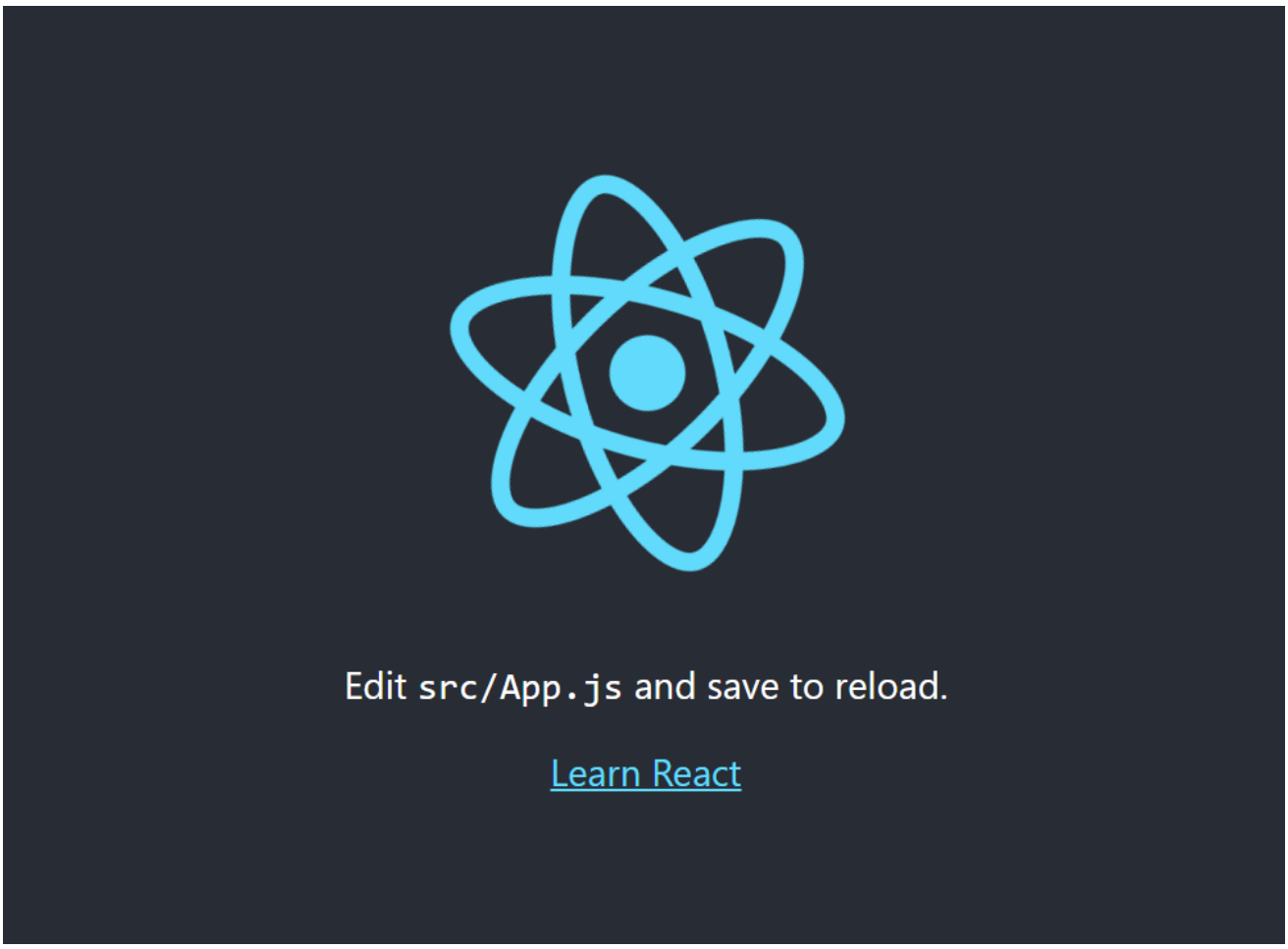


Рисунок 4.7 – *встановлення React додатку*

4.3 Розробка макету web-додатку

Після проведення аналізу аналогів інших web-додатків створюємо макет майбутнього web-додатку з допомоги сайту draw io.

Спочатку створимо макет головної сторіки, які побачить користувач коли зайде на сторінку нашого додатку, на якій буде відображено, що наш web-додаток створений для проведення опитувань, та буде кнопка входу через Google (рис 4.8).

Потім створюємо макет відображення в нашому web-додатку, де буде зрозуміло коли було надіслано опитувань, скільки людей відповіло «Так» чи «Ні», конпка для видалення опитування та створення нового опитування (Рис 4.9).

Створюжмо макет сторінки для створення нових опитувань, в якому вказуємо: назву компанії, тему питання, текст питання, та список одержувачів, також 2 кнопки «Скасувати» та «Далі» (Рис 4.10).

Останній макет перевірка введених даних в опитування та надсилання опитування для користувачі, на якій є можливість повернутись назад та відредагувати данні кнопкою «Назад» або надіслати опитування кнопкою «Надіслати опитування» (Рис 4.11).

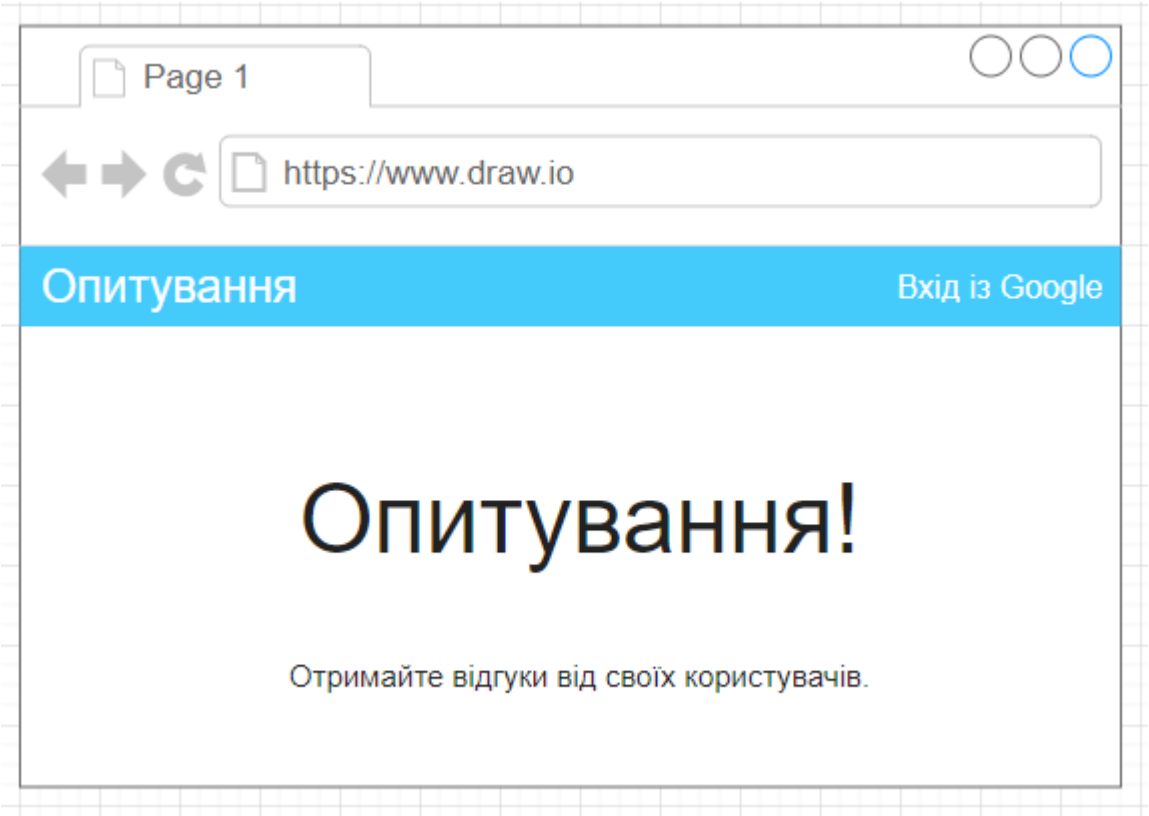


Рисунок 4.8 – *макет головної сторінки web-додатку*

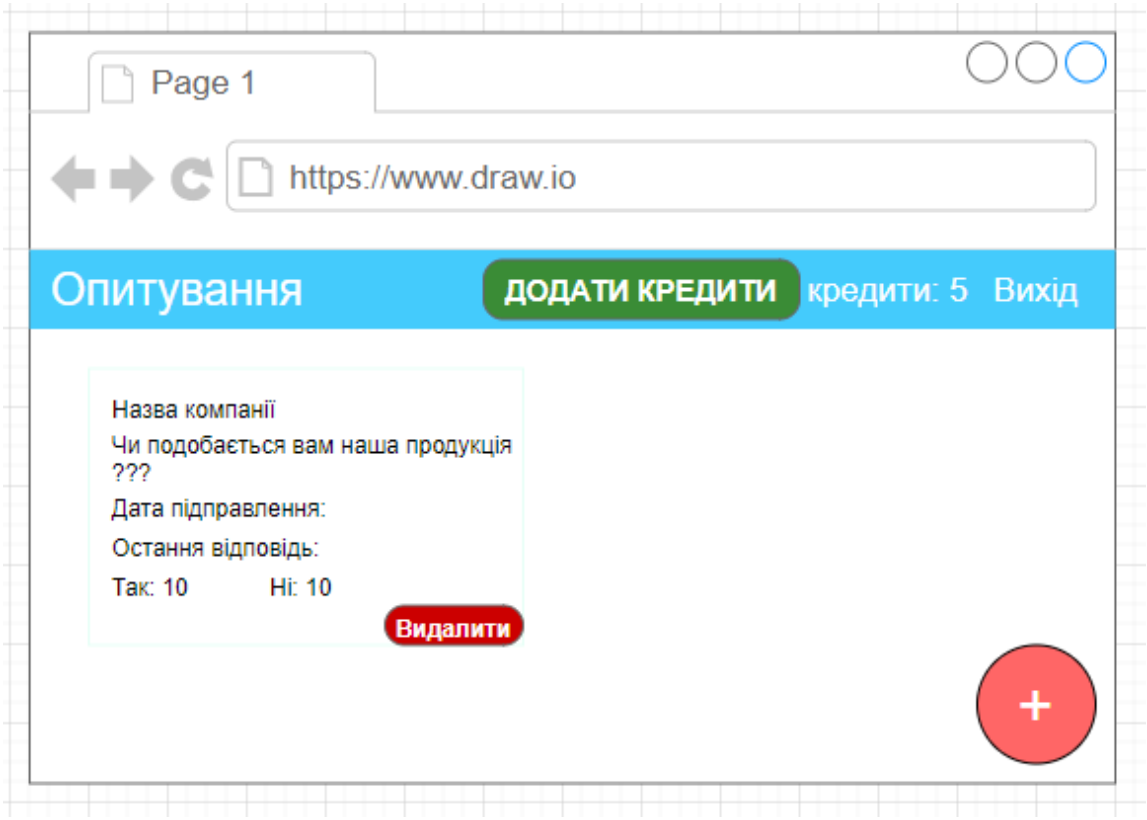


Рисунок 4.9 – *макет відображення опитувань web-додатку*

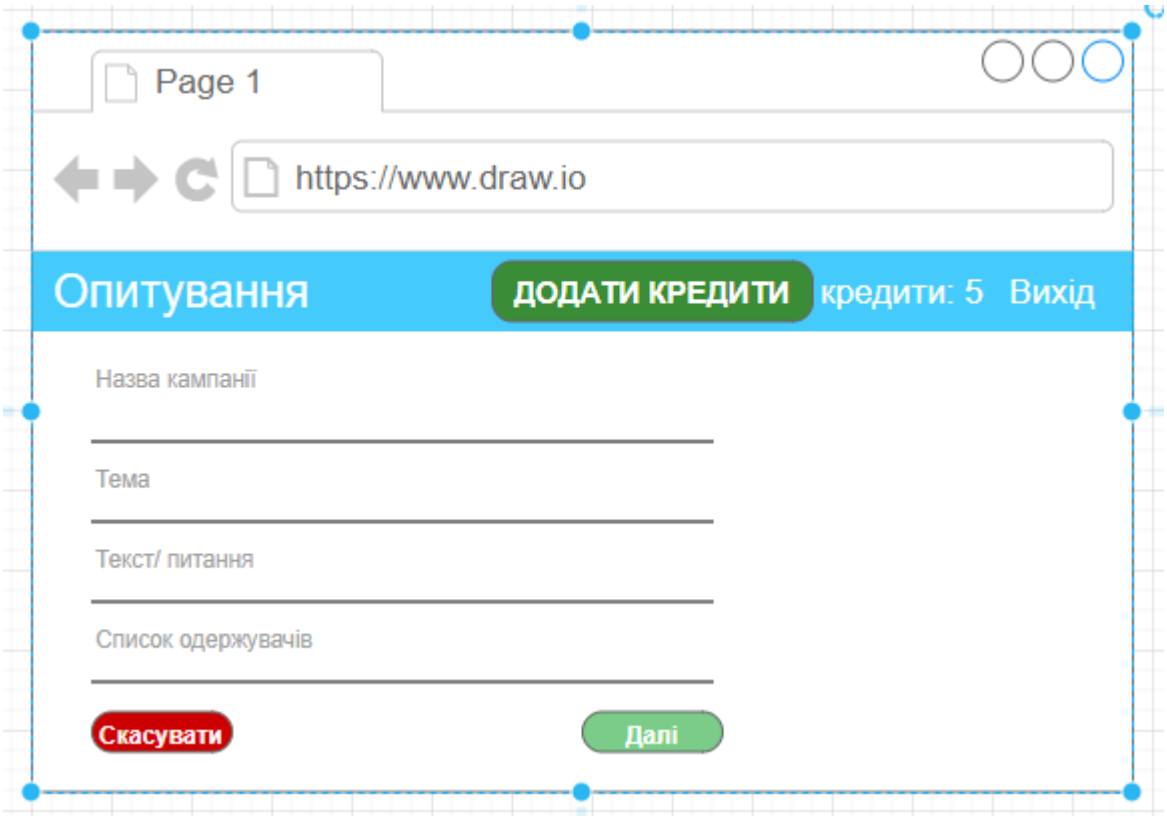


Рисунок 4.10 – *макет створення опитувань web-додатку*

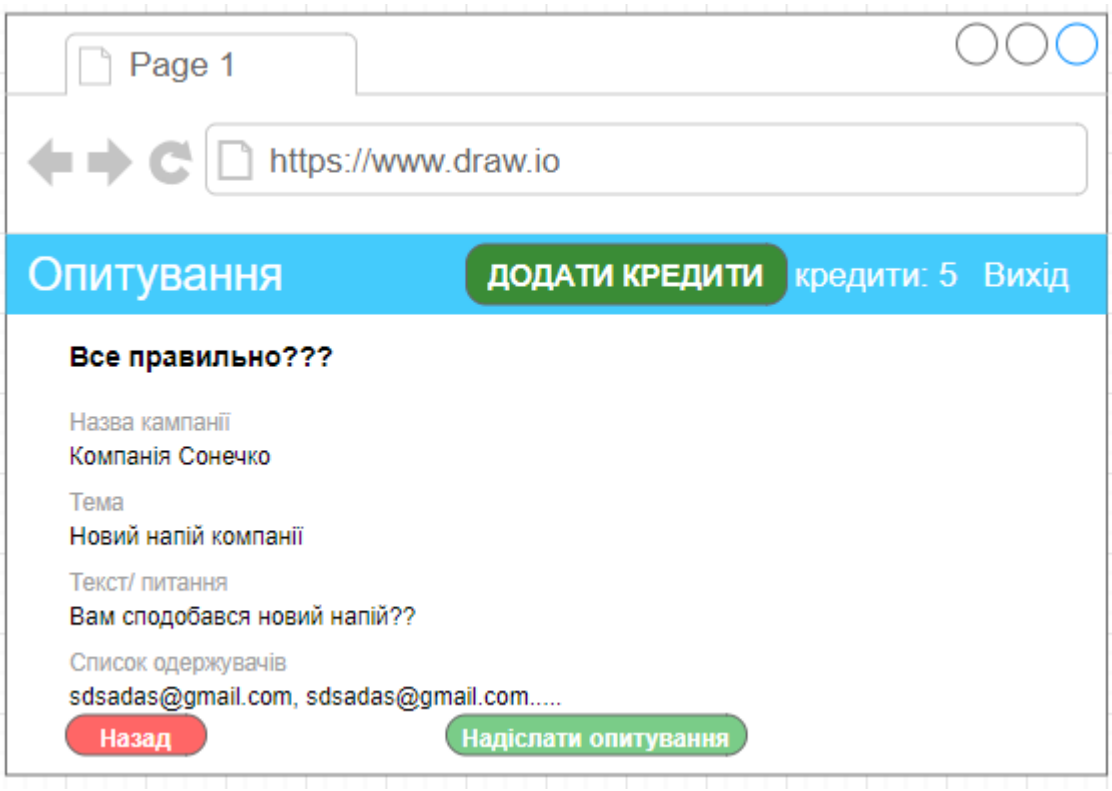


Рисунок 4.11 – *макет підтвердження опитувань web-додатку*

4.4 Встановлення та підключення mongodb

Для роботи додатків React потрібно зберігати отримані дані, та зручно використовувати, з цим на допоможе сайт https://www.mongodb.com/ (рис 4.12)

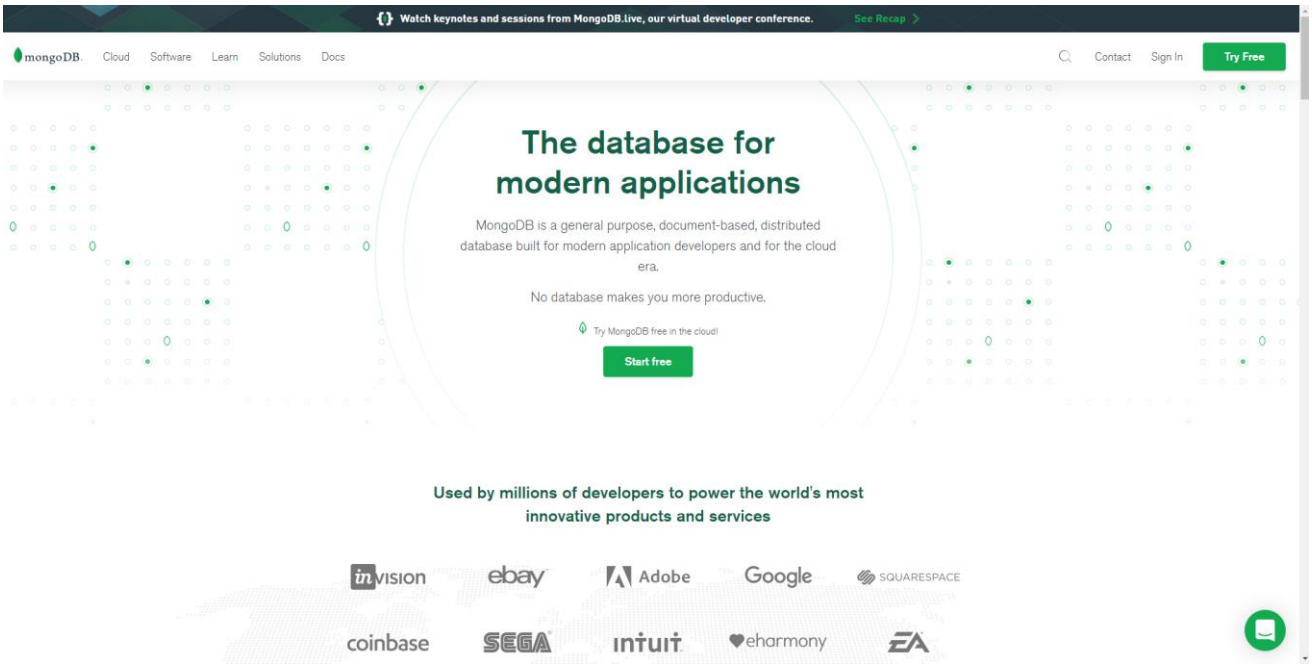


Рисунок 4.12 – *головна сторінка сайту mongodb*

Для встановлення та підключення потрібно встановити бібліотеку mongodb в наш додаток, з допомоги команди npm install mongodb –save.[7]

Після встановлення бібліотеки, переходимо до створення бази даних на сайті mongodb (рис 4.13), та отримуємо ключ для підключення бази даних web-додатку (рис 4.14).

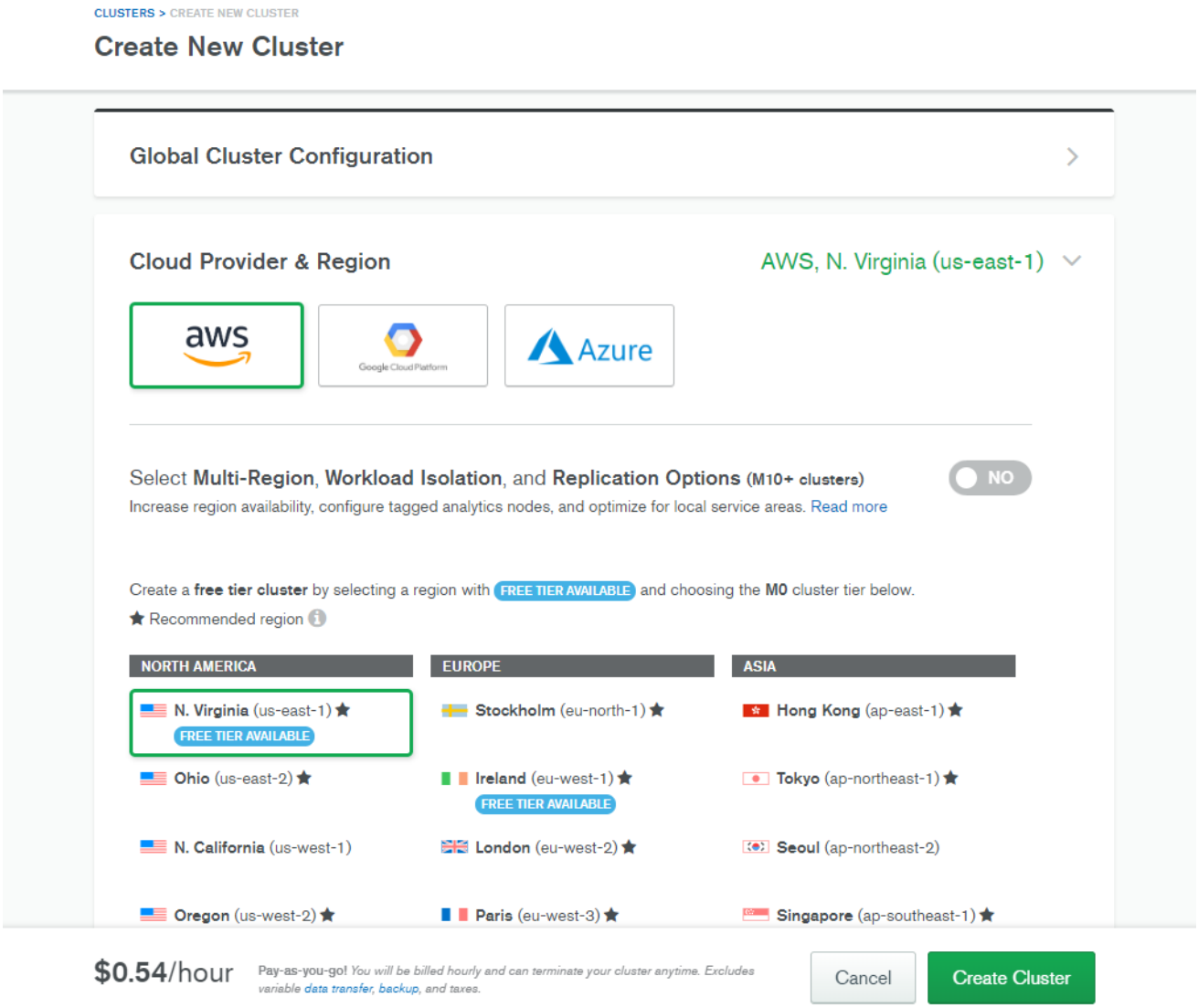


Рисунок 4.13 – *створення бази даних mongodb*

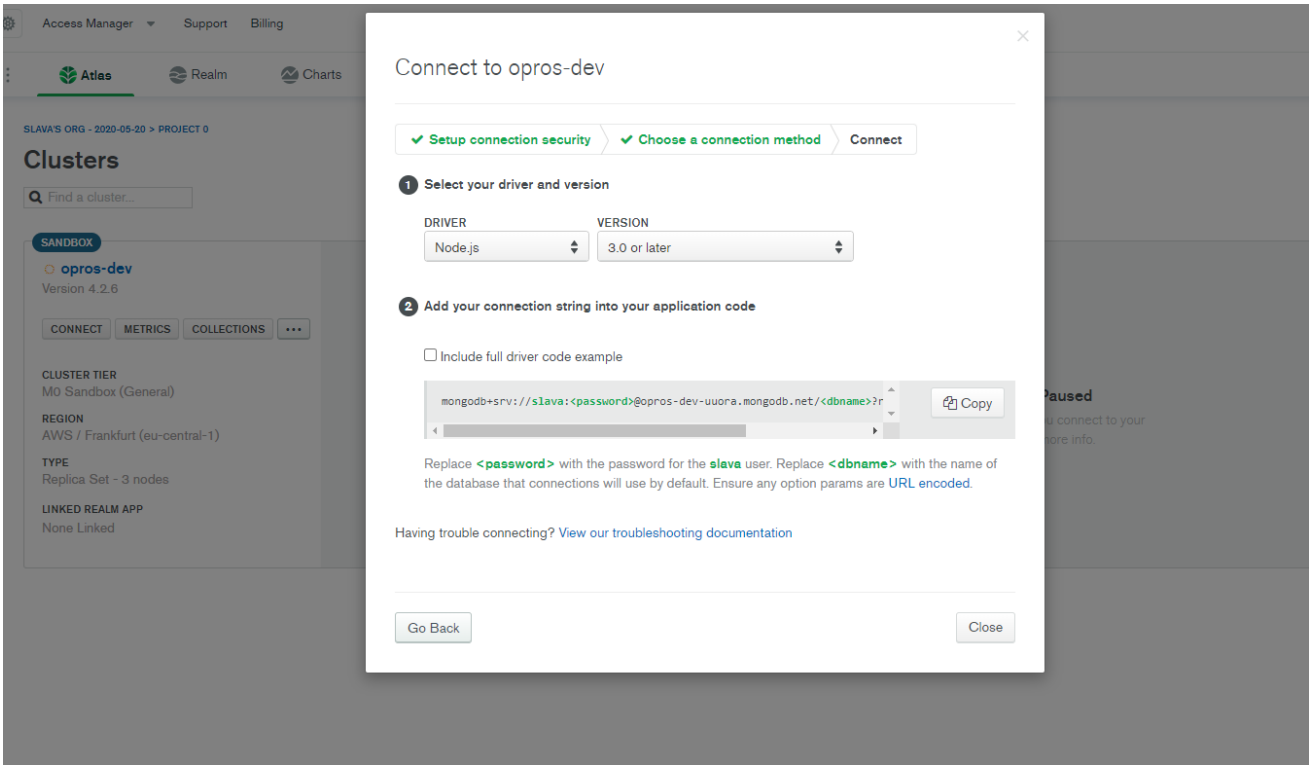


Рисунок 4.14 – *підключення mongodb*

База даних mongodb автоматично зберігає дані які надсилає web-додаток і добре підтримує node js, що для нас важливо.

4.5 Підключення API

Для роботи web-додатку потрібно реалізувати і підключити основні API.

SendGrid додаток для надсилання повідомлень, з його допомогою web-додаток буде відправляти та відстежувати повідомлення надіслані користувачам для проведення опитувань, головна сторінка сайду (рис 4.15), отримання API для підключення (рис 4.16).[6]

Stripedocs додаток для реалізації оплати, з допомого. Якого працюють картки в тестовому режимі, головна сторінка сайду (рис 4.17), отримання API для підключення (рис 4.18).[5]

Google API для реалізації авторизації в web-додатку, швидко та звучно надає мозливисть створючати акант в додатку та користуватись ним, головна сторінка сайду (рис 4.19), отримання API для підключення (рис 4.20).[4]

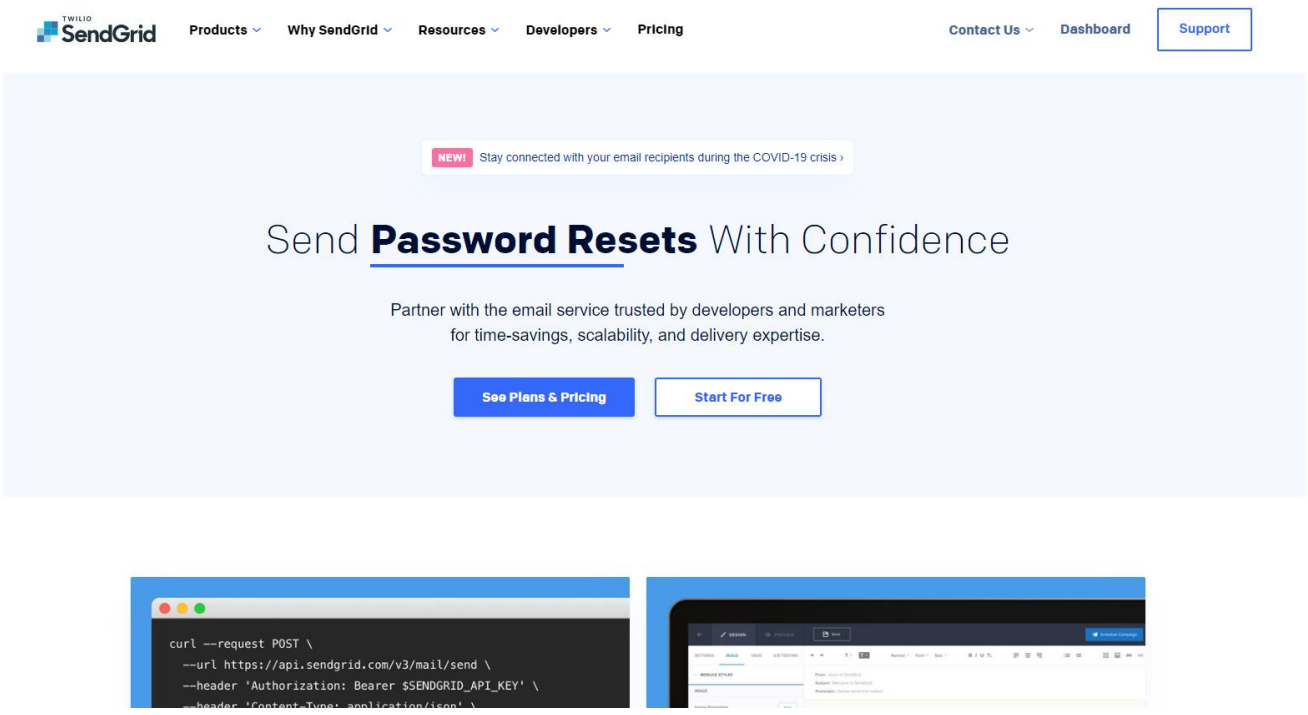


Рисунок 4.15 – *головна сторінка сайту SendGrid*

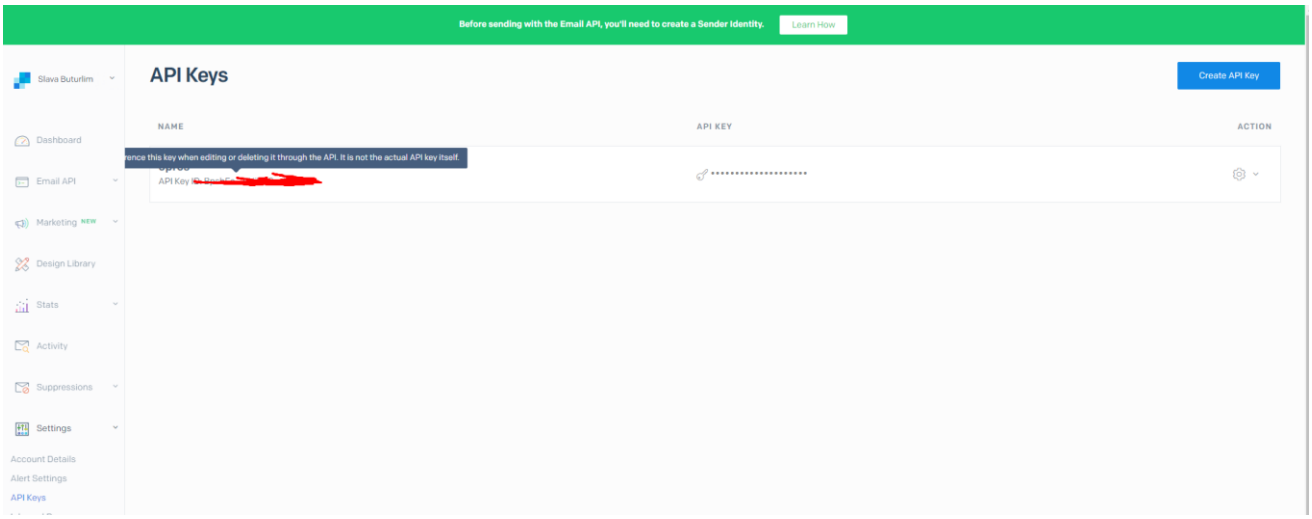


Рисунок 4.16 – *отримання кодів для підключення SendGrid*

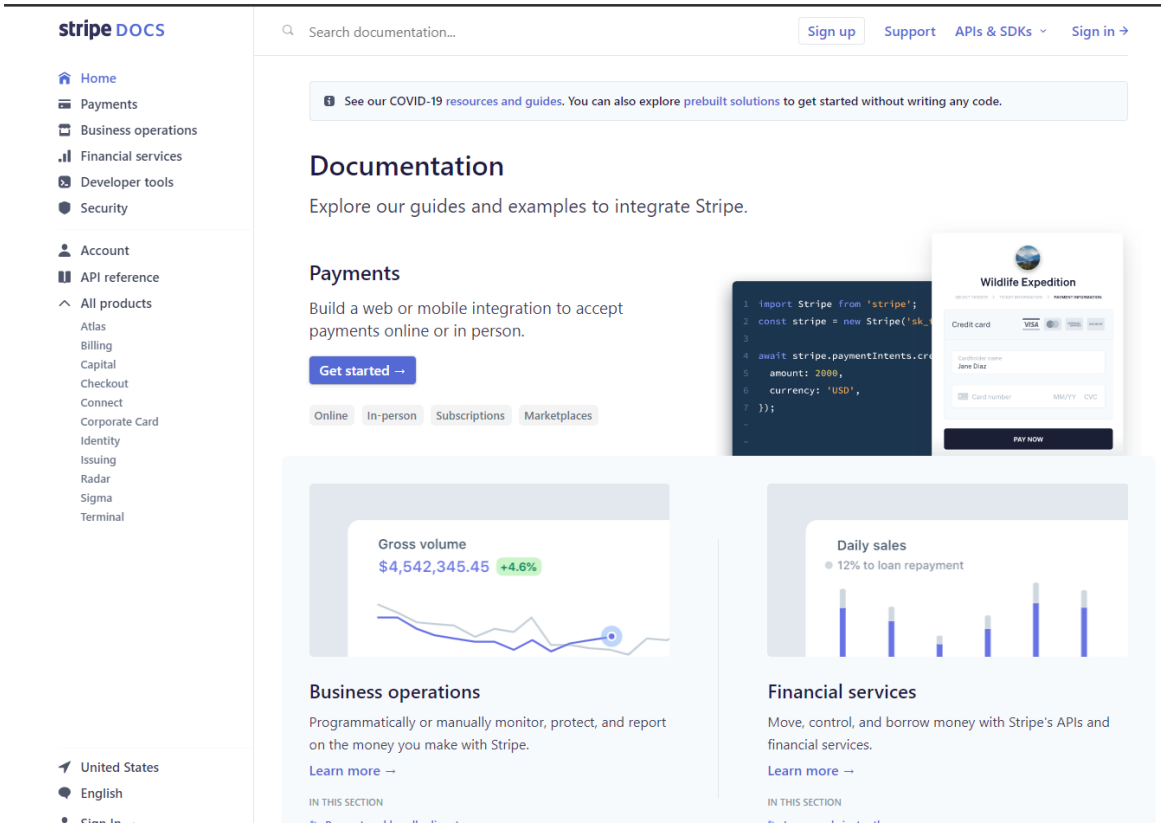


Рисунок 4.17 – *головна сторінка stripedocs*

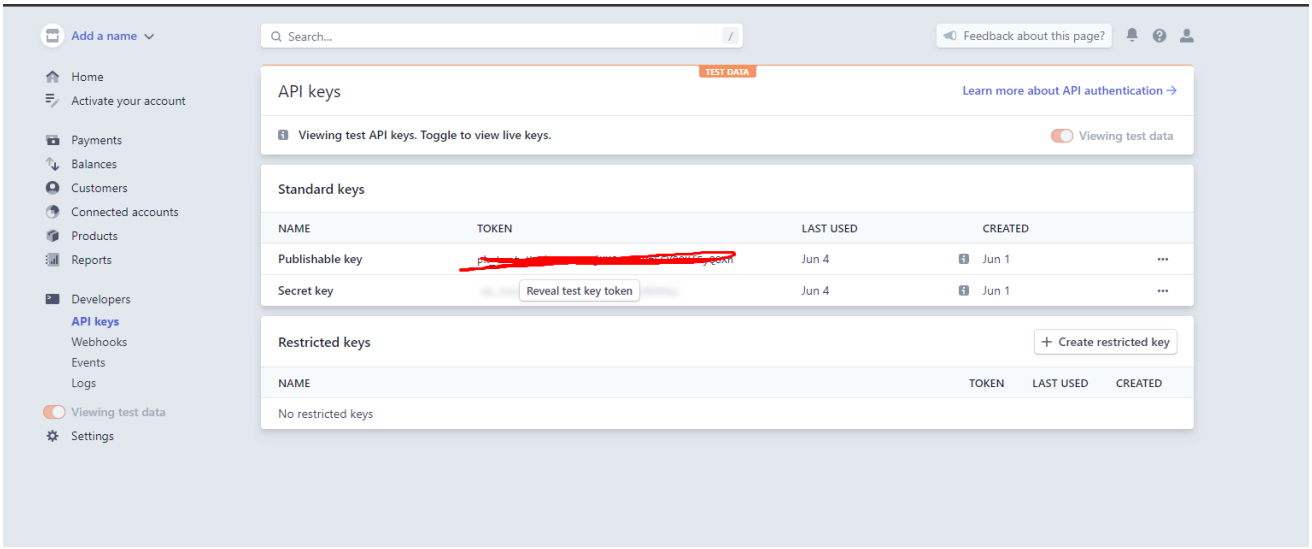


Рисунок 4.18 – *отримання кодів для підключення stripedocs*

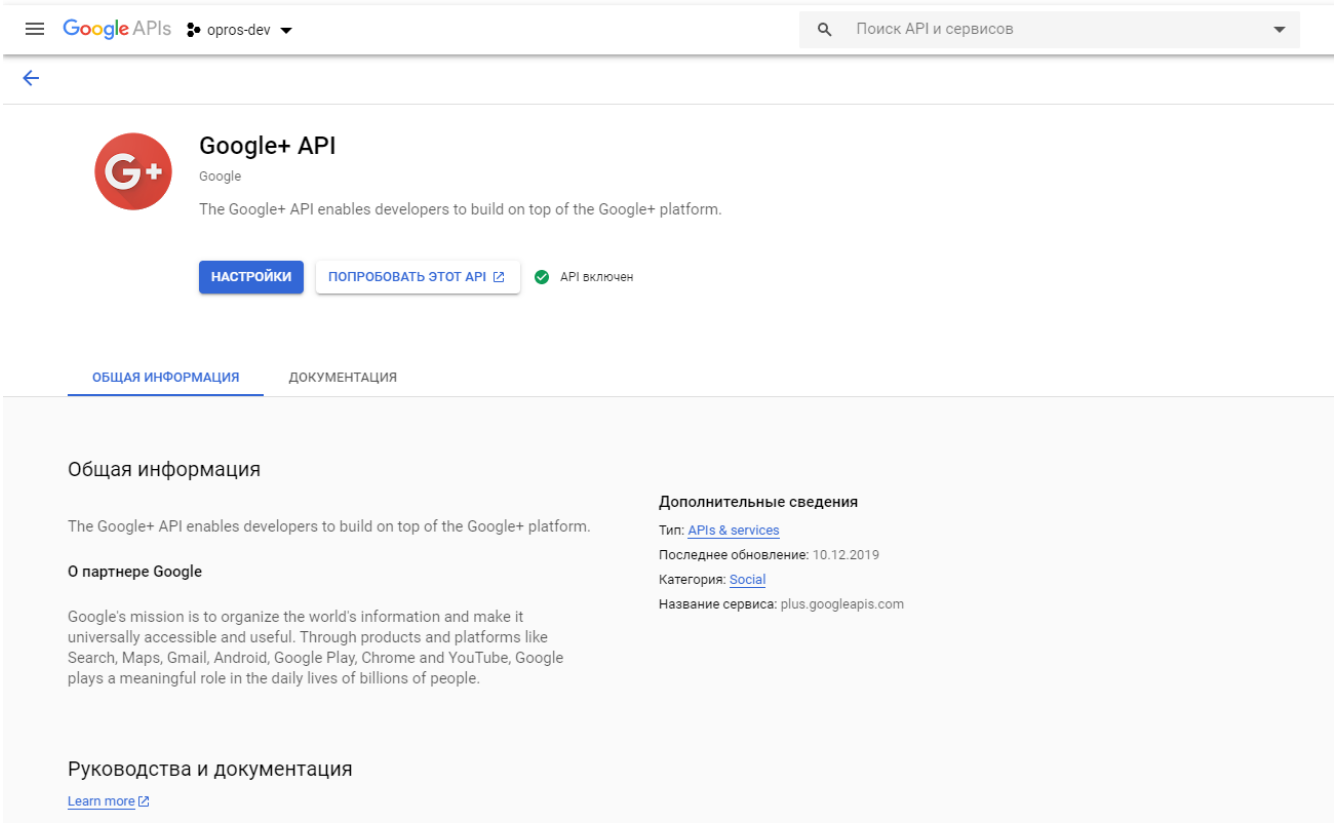


Рисунок 4.19 – *головна сторінка Google API*

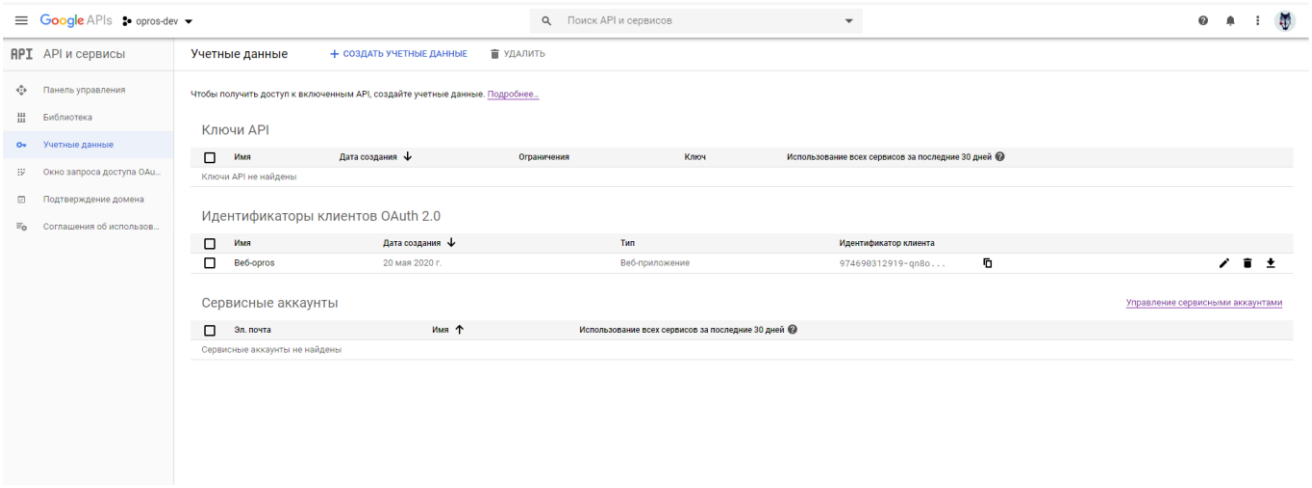


Рисунок 4.20 – *отримання кодів для підключення Google*

4.6 Створення web-додатку на heroku

Heroku — хмана платформа для web-додатків які підтримує широкий ряд мов програмування, одна із перших хмарних платформ які з’явились в мережі Інтернет заснована команією Heroku працює на операційній системі Debian підтримує мови Node.js, Clojure, Python і PHP та інші.[3]

Головна сторінка сайту сайту для створення web-додатків (рис. 4.21).

Для встановлення бібліотеки в наш додаток потрібно прописати команду npm install -g heroku, після цього провести авторизацію і підключити всі основні ключі в параметри програми (рис. 4.22-23).

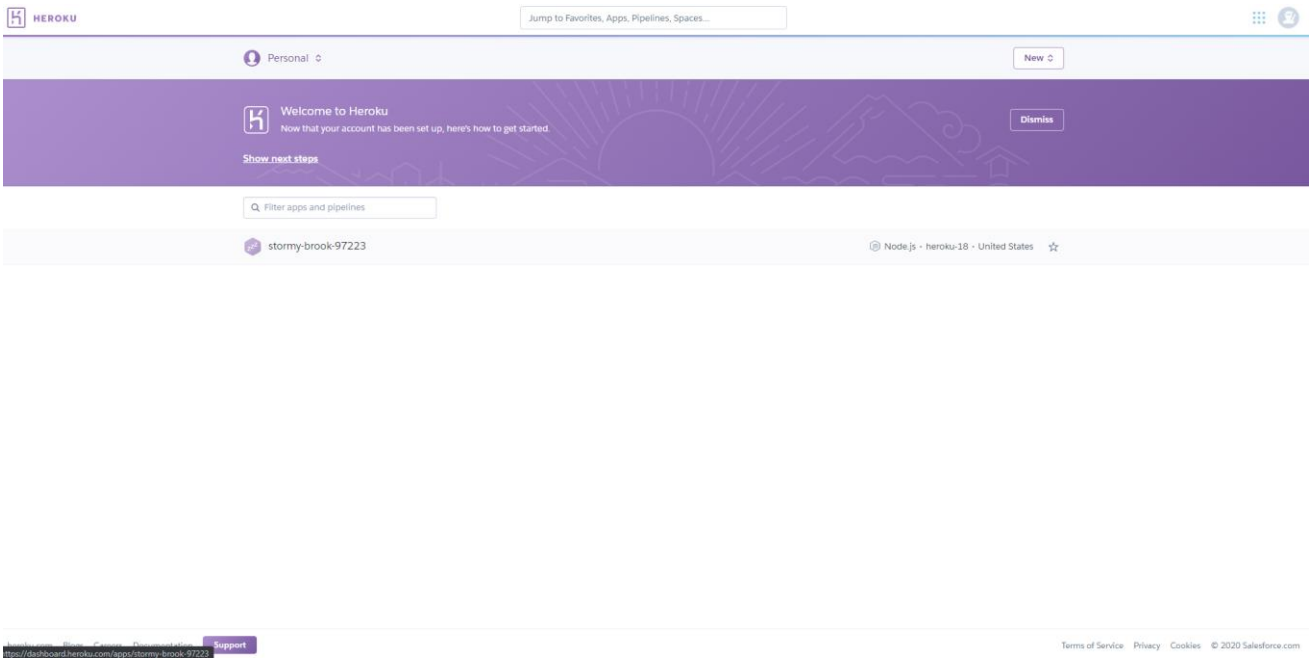


Рисунок 4.21 – *головна сторінка Heroku*

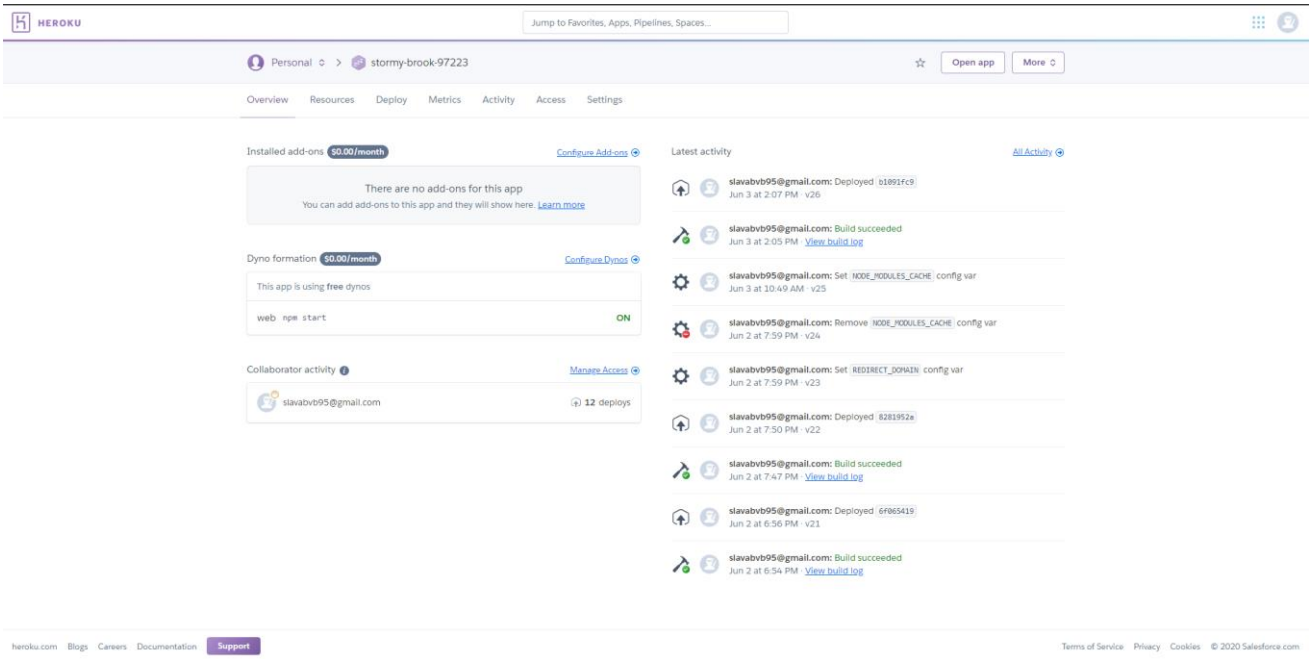


Рисунок 4.22 – *параметри web-додатку*

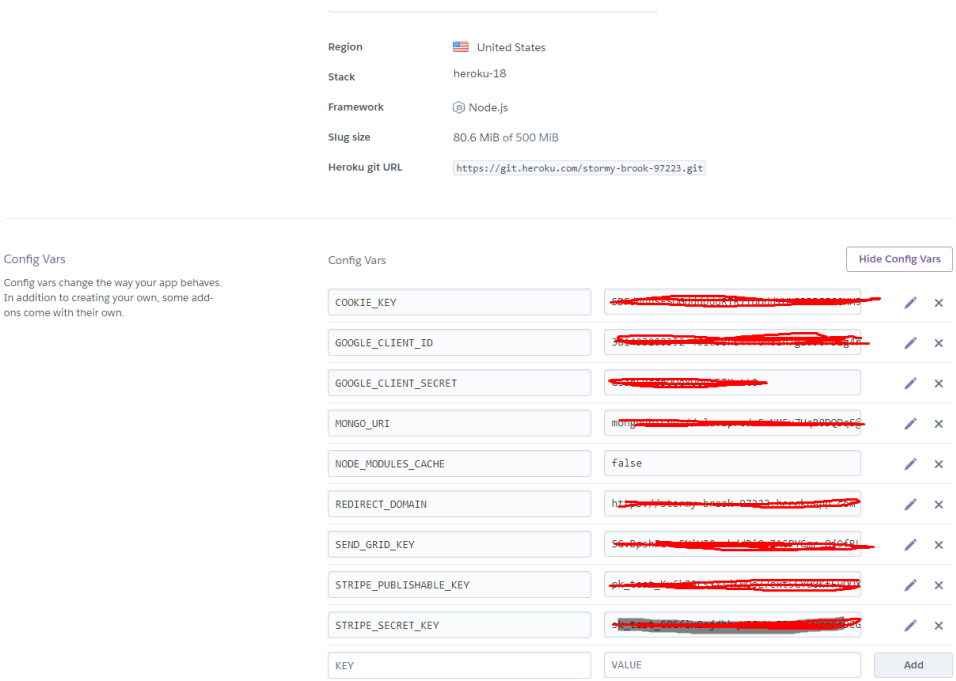


Рисунок 4.23 – *основні ключі web-додатку*

4.7 Написання основного коду

Після розробки макету додатку було розроблено сторінки web-додатку та написано основний код для них.

Реалізований зручний вхід з допомогою системи Google. Написаний основний код для авторизації з допомоги Google (рис 2.24).

Розроблено головну сторінку на якій є кнопка входу через Google (рис 2.26), та написаний код головної сторінки (рис 2.25).

Розроблено головну сторінку для додавання опитувань де створена кнопка для додавання опитувань (рис 2.28), та написаний код для неї (рис 2.27).

Розроблено головну сторінку для створення опитувань з головними полями які необхідні при створенні опитувань, тема опитування, назва компанії яка проводить опитування та кому надсилається опитування, реалізовано перевірку полів, бою не допустити пустих полів (рис 2.30), та написаний код для неї (рис 2.29).

Розроблено головну сторінку для перевірки правильності введення даних для опитування (рис 2.32), та написаний код для неї (рис 2.31).

Розроблено сторінку для відображення опитувань на якій добре видно дату відправлення опитування, коли була остання відповідь ті кількість відповідей по опитуванню (рис 2.34), та написаний код для неї (рис 2.33).

Вигляд повідомлення опитування яке надходить на почту (рис 2.36), та написаний код створення повідомлення (рис 2.35).

Написаний код для перевірки правильності введення електронної адреси (рис 2.37).

```
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const mongoose = require('mongoose');
const keys = require('../config/keys');

const User = mongoose.model('users');

passport.serializeUser((user, done) => {
  done(null, user.id);
});

passport.deserializeUser((id, done) => {
  User.findById(id).then(user => {
    done(null, user);
  });
});

passport.use(
  new GoogleStrategy(
    {
      clientID: keys.googleClientID,
      clientSecret: keys.googleClientSecret,
      callbackURL: '/auth/google/callback',
      proxy: true
    },
    async (accessToken, refreshToken, profile, done) => {
      const existingUser = await User.findOne({ googleId: profile.id });

      if (existingUser) {
        return done(null, existingUser);
      }

      const user = await new User({ googleId: profile.id }).save();
      done(null, user);
    }
  )
);
```

Рисунок 4.24 – *основний код авторизації*

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { Link } from 'react-router-dom';
import Payments from './Payments';

class Header extends Component {
  renderContent() {
    switch (this.props.auth) {
      case null:
        return;
      case false:
        return <li><a href="/auth/google">Вхід із Google</a></li>;
      default:
        return [
          <li key="1"><Payments /></li>,
          <li key="3" style={{ margin: '0 10px' }}>
            Кредити: {this.props.auth.credits}
          </li>,
          <li key="2"><a href="/api/logout">Вийти</a></li>
        ];
    }
  }

  render() {
    return (
      <nav style={{ backgroundColor: '#6eb9ee' }}>
        <div className="nav-wrapper">
          <Link
            to={this.props.auth ? '/surveys' : '/'}
            className="left brand-logo"
          >
```

Рисунок 4.25 – *основний код головної сторінки web-додатку*

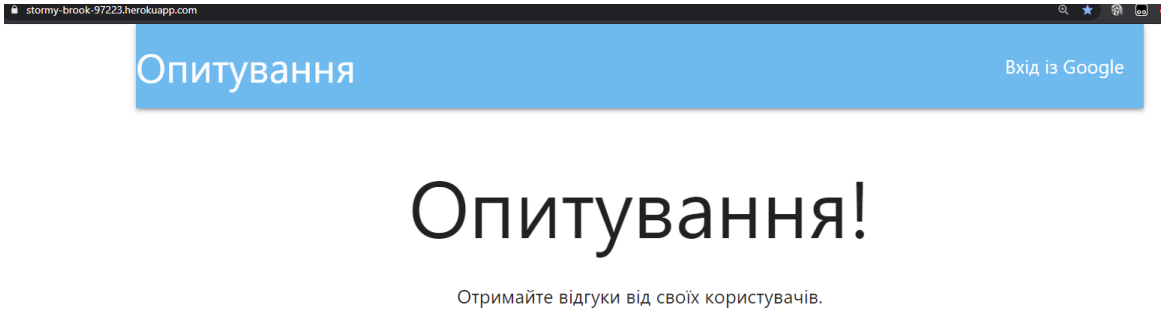


Рисунок 4.26 – *головна сторінка web-додатку*

client > src > components > JS Dashboard.js > ...

```
1   import React from 'react';
2   import { Link } from 'react-router-dom';
3   import SurveyList from './surveys/SurveyList';
4
5
6   const Dashboard = () => {
7     return (
8       <div>
9         <SurveyList />
10        <div className="fixed-action-btn">
11          <Link to="/surveys/new" className="btn-floating btn-large red">
12            <i className="material-icons">add</i>
13          </Link>
14        </div>
15      </div>
16    );
17  };
18
19
20  export default Dashboard;
21
```

Рисунок 4.27 – *основний код додавання опитувань*

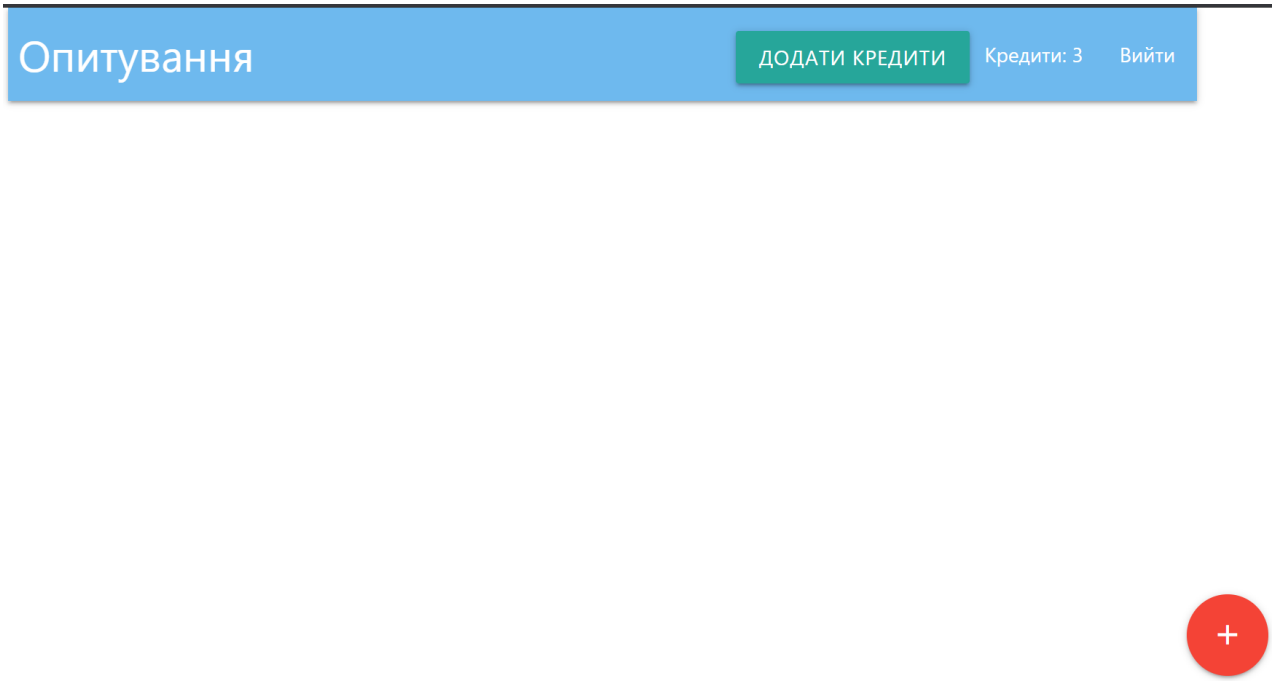


Рисунок 4.28 – *відображення сторінки додавання опитувань*

```
import _ from 'lodash';
import React, { Component } from 'react';
import { reduxForm, Field } from 'redux-form';
import { Link } from 'react-router-dom';
import SurveyField from './SurveyField';
import validateEmails from '../../utils/validateEmails';
import formFields from './formFields';

class SurveyForm extends Component {
  renderFields() {
    return _.map(formFields, ({ label, name }) => {
      return (
        <Field
          key={name}
          component={SurveyField}
          type="text"
          label={label}
          name={name}
        />
      );
    });
  }

  render() {
    return (
      <div>
        <form onSubmit={this.props.handleSubmit(this.props.onSurveySubmit)}>
          {this.renderFields()}
          <Link to="/surveys" className="red btn-flat white-text">
          Скасувати
          </Link>
          <button type="submit" className="teal btn-flat right white-text">
          Далі
            <i className="material-icons right">done</i>
          </button>
        </form>
      </div>
    );
  }
}

function validate(values) {
  const errors = {};

  errors.recipients = validateEmails(values.recipients || '');
```

Рисунок 4.29 – *код створення опитування*

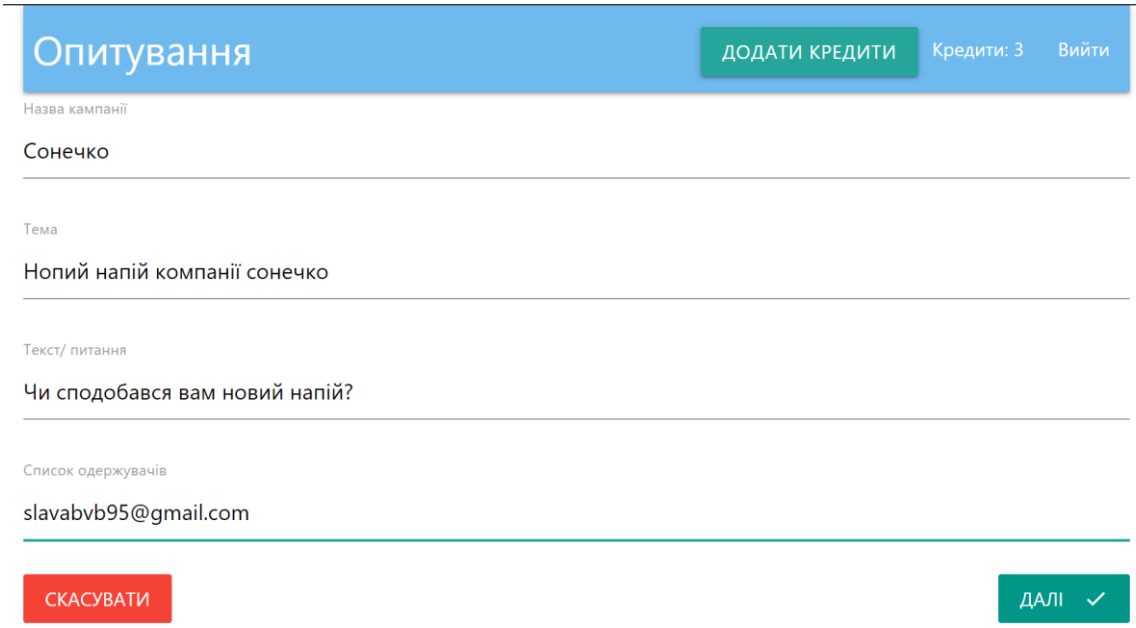


Рисунок 4.30 – *вигляд сторінки для створення опитувань web-додатку*

```
import _ from 'lodash';
import React from 'react';
import { connect } from 'react-redux';
import formFields from './formFields';
import { withRouter } from 'react-router-dom';
import * as actions from '../../actions';

const SurveyFormReview = ({ onCancel, formValues, submitSurvey, history }) => {
  const reviewFields = _.map(formFields, ({ name, label }) => {
    return (
      <div key={name}>
        <label>{label}</label>
        <div>
          {formValues[name]}
        </div>
      </div>
    );
  });

  return (
    <div>
      <h5>Все правильно?</h5>
      {reviewFields}
      <button
        className="yellow darken-3 white-text btn-flat"
        onClick={onCancel}
      >
        Назад
      </button>
      <button
        onClick={() => submitSurvey(formValues, history)}
        className="green btn-flat right white-text"
      >
        Надіслати опитування
        <i className="material-icons right">email</i>
      </button>
    </div>
  );
};

function mapStateToProps(state) {
  return { formValues: state.form.surveyForm.values };
}
```

Рисунок 4.31 – *код сторінки перевірки опитування та надсилання*

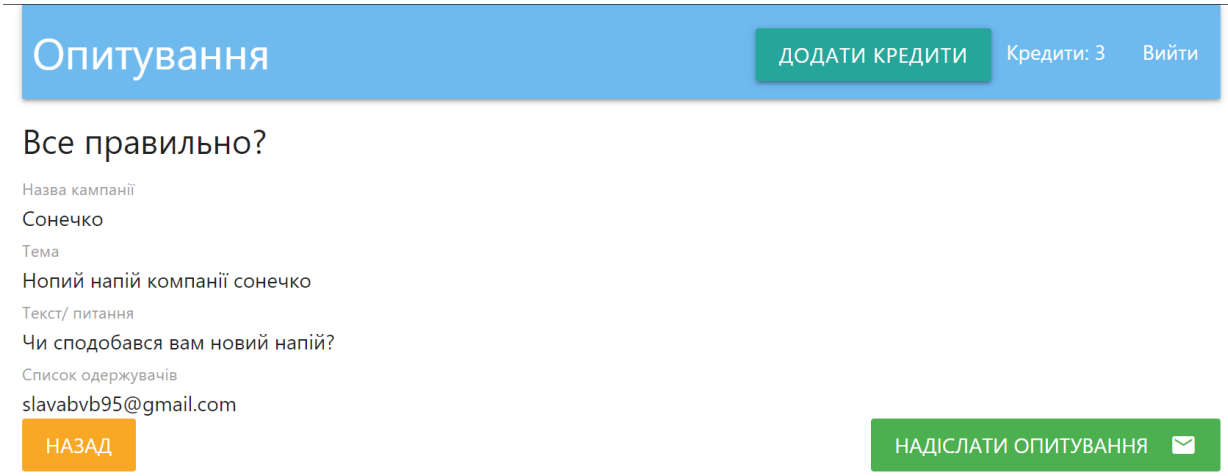


Рисунок 4.32 – *відображення сторінки надсилання опитувань web-додатку*

```jsx
import React from 'react';
import { connect } from 'react-redux';
import { fetchSurveys, deleteSurvey } from '../../actions';

class SurveyList extends React.Component {

    componentDidMount() {
        this.props.fetchSurveys();
    }

    renderSurveys() {
        return this.props.surveys.reverse().map(survey => {
            return (
                <div className="card darken-1" key={survey._id}>
                    <div className="card-content">
                        <span className="card-title">{survey.title}</span>
                        <p>{survey.body}</p>
                        <p className="right">Дата відправлення: {new Date(survey.dateSent).toLocaleDateString()}</p>
                        <br />
                        <p className="right">Остання відповідь:
                            {survey.lastResponded ? new Date(survey.lastResponded).toLocaleDateString() : 'Немає відповідей'}
                        </p>
                    </div>
                    <div className="card-action">
                        <span style={{ marginRight: '20px', color: 'green' }}>Так: {survey.yes}</span>
                        <span style={{ color: 'red' }}>Ні: {survey.no}</span>
                        <button
                            className="red right btn-flat white-text"
                            style={{ marginTop: '-5px' }}
                            onClick={() => { this.props.deleteSurvey(survey._id) }}
                        >Видалити</button>
                    </div>
                </div>
            );
        })
    }

    render() {
        return (
            <div>{this.renderSurveys()}</div>
        );
    }
}

function mapStateToProps({ surveys }) {
```

Рисунок 4.33 – *код відображення опитувань*

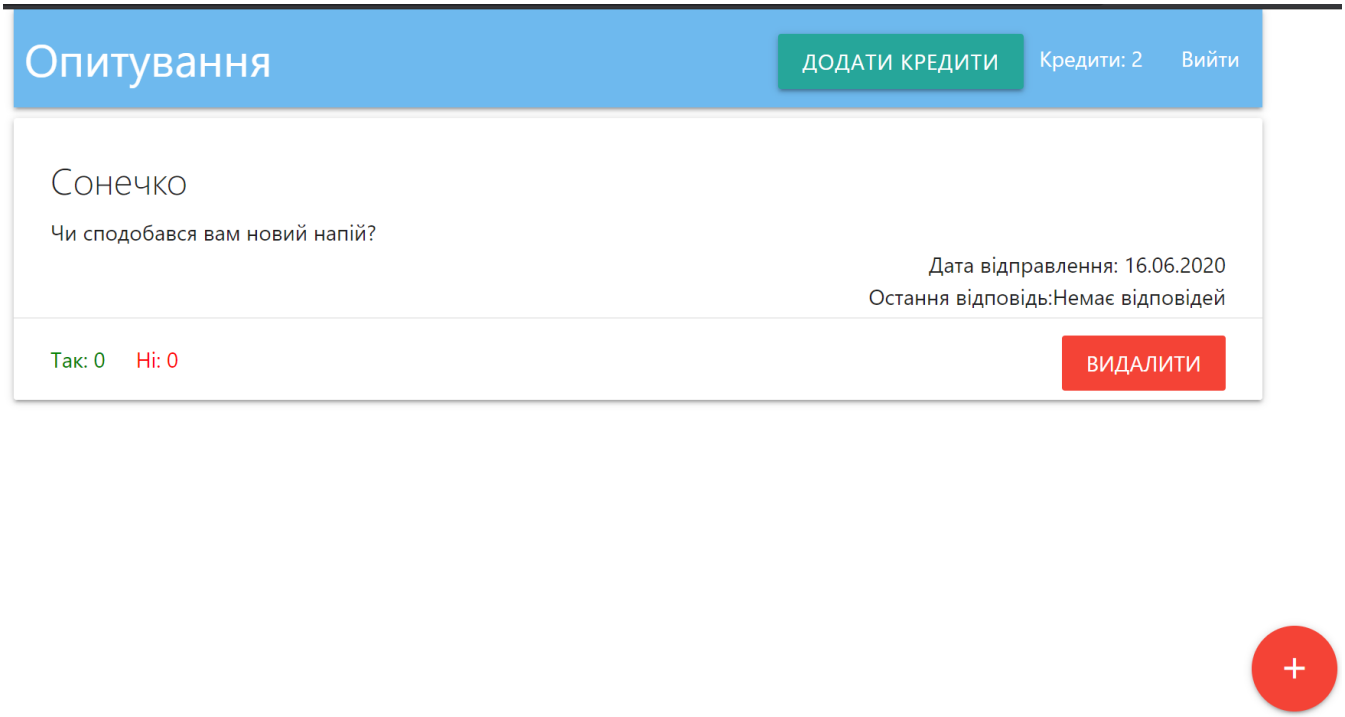


Рисунок 4.34 – *сторінка відображення опитувань web-додатку*

```
const keys = require('../../config/keys');

module.exports = survey => {
  return `
    <html>
      <body>
        <div style="text-align: center;">
          <h3>Зробіть свій вибір!</h3>
          <p>Будь ласка, дайте відповідь на наступне запитання:</p>
          <p>${survey.body}</p>
          <div>
            <a href="${keys.redirectDomain}/api/surveys/${survey.id}/yes">Так</a>
          </div>
          <div>
            <a href="${keys.redirectDomain}/api/surveys/${survey.id}/no">Ні</a>
          </div>
        </div>
      </body>
    </html>
  `;
};
```

Рисунок 4.35 – *код повідомлення*

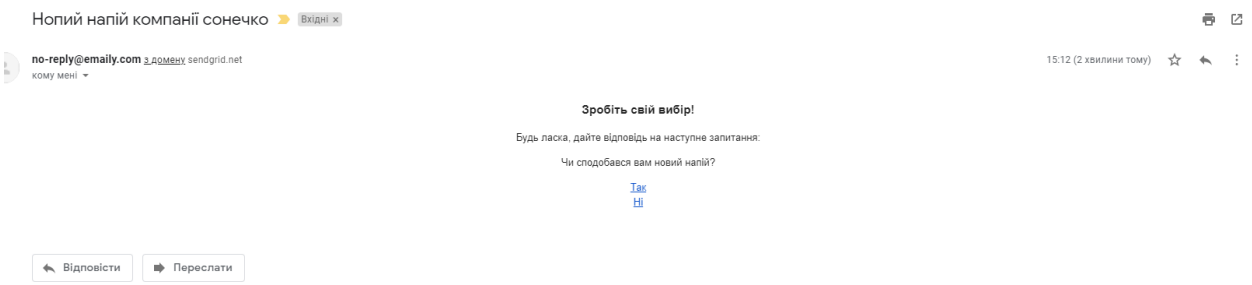


Рисунок 4.36 – *вигляд отриманого опитування*

```
const re = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;

export default emails => {
  const invalidEmails = emails
    .split(',')
    .map(email => email.trim())
    .filter(email => re.test(email) === false);

  if (invalidEmails.length) {
    return `Недійсна електронна пошта: ${invalidEmails}`;
  }

  return;
};

```

Рисунок 4.37 – *код перевірки email*

5. ТЕСТУВАННЯ WEB-ДОДАТКУ ДЛЯ ПРОВЕДЕННЯ МАРКЕТИНГОВИХ ДОСЛІДЖЕНЬ

Останнім кроком розробки web-додатку - є тестування для знаходження помилок, та для перевірки якості розробленого web-додатку для проведення маркетингових досліджень. Головною якістю повинен бути готовий продукт який не має помилок, має зручний інтерфейс та повинен бути простим у використанні.

Було проведено тестування розробленого web-додатку для проведення маркетингових досліджень (таблиця 5.1).

Таблиця 5.1 – Тестування web-додатку для проведення маркетингових досліджень

Номер тестування	Сценарій тестування	Дія	Результат	Статус
Тест 1.	Перехід на головну сторінку web-додатку	1. Відкривання браузеру 2. Ведення адреси web-додатку 3. Перехід на сторінку web-додатку	Перехід на головну сторінку web-додатку	Відбуваєтьс я
Тест 2.	Авторизація через Google	1. Натискання кнопки «Вхід із Google» 2. Ведення даних від акаунту Google 3. Натискання кнопки «Далі»	Успішна авторизація	Відбуваєтьс я
	Створення опитування	1. Натискання кнопки для створення опитувань «+» 2. Ведення даних у поле вводу «Назва кампанії» 3. Ведення даних у поле вводу «Тема» 4. Ведення даних у поле вводу «Текст/ питання» 5. Ведення даних у поле вводу «Список одержувачів» 6. Натискання кнопки «Далі»	Успішне введення даних в поля введення, перехід на форму підтверджен ня відправленн я опитування	Відбуваєтьс я

Продовження таблиці – Тестування web-додатку для проведення маркетингових досліджень

Тест 4.	Відправлення опитування	1. Натискання кнопки «НАДІСЛАТИ ОПИТУВАННЯ»	Успішне відправлення опитування, та перехід до списку опитувань	Відбуваєтьс я
Тест 5.	Проходження опитування	1. Відкривання електронного почтового повідомлення з опитування 2. Натискання гіперпосилання «Так»	Успішне проведення опитування	Відбуваєтьс я

Під час проведення тестування помилок в ході роботи web-додатку для проведення маркетингових досліджень не виявлено, всі функції працюють добре.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було проведено дослідження та проаналізовано всі головні інформаційних потоки, які працюють у сфері маркетингових досліджень методом опитувань. З допомогою аналізу аналогів було знайденні недоліки та переваги, які надали змогу в ході проектуванні web-додатку зробити його якомога зручнішим.

Результатом кваліфікаційної роботи бакалавра є web-додаток, який дає змогу створювати прості опитування, відображання результатів опитування. Розроблений зручний дизайн.

Для розробки web-додатку було проведено:

- пошук та аналіз інформації згідно щодо застосування інформаційних технологій у маркетингових дослідженнях;
- визначення переваг та недоліків подібних систем;
- обрані мови програмування та web-додатки вирішення поставленої мети дипломного проектування;
- створена структура завдань у вигляді діаграми Ганта з часовою шкалою,
- розроблено дизайн web-додатку,
- розроблено Web-додаток для проведення маркетингових досліджень,
- проведено тестування web-додатку та перенесено на хостинг,
- реалізовано авторизацію через Google.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оліщук Андрій Володимирович. Розробка Web-додатків на PHP 5. Професійна робота. — М. : «Вільямс», 2006. — С. 352. (дата звернення: 15.05.20)

2. Фрэд Лонг та ін.. Руководство для программиста WEB / «Вільямс». – К.: Каліпсо, 2014 (дата звернення: 15.05.20)

3. Коротке описання ресурсу Heroku [Електронний ресурс] wikipedia 2020 - Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/Heroku (дата звернення: 10.05.20)

4. Метод підключення Google API [Електронний ресурс] Google 2020 - Режим доступу до ресурсу: https://console.developers.google.com/ (дата звернення: 11.05.20)

5. Голона сторінка сайту Stripedocs [Електронний ресурс] Stripe 2020 - Режим доступу до ресурсу: https://stripe.com/ (дата звернення: 11.05.20)

6. Головна сторінка сайту SendGrid [Електронний ресурс] SendGrid 2020 - Режим доступу до ресурсу: https://sendgrid.com/ (дата звернення: 12.05.20)

7. Метод підключення і роботи з базою данних Mongodb [Електронний ресурс] Mongodb 2020 - Режим доступу до ресурсу: https://docs.mongodb.com/drivers/node/ (дата звернення: 17.05.20)

8. Метод створення React додатків Create React App [Електронний ресурс] Github 2020 - Режим доступу до ресурсу: https://github.com/facebook/create-react-app (дата звернення: 09.05.20)

9. Описання Node js [Електронний ресурс] Node js 2020 - Режим доступу до ресурсу: https://nodejs.org/uk/about/ (дата звернення: 08.05.20)

10. Описання программи Visual Studio Code [Електронний ресурс] Wikipedia 2020 - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 07.05.20)

11. Діаграма прецедентів Вікіпедія [Електронний ресурс] Wikipedia 2020 - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 15.02.20)

12. Офіціальна сторінка React [Електронний ресурс] Reactjs 2020 - Режим доступу до ресурсу: https://uk.reactjs.org/ (дата звернення: 11.05.20)

13. JavaScript. Вікіпедія [Електронний ресурс] Wikipedia 2020 - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript (дата звернення: 15.02.20)

14. Сторінка з веб-сайту Маркетингове дослідження. Вікіпедія [Електронний ресурс] Wikipedia 2020 - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Маркетингове_дослідження (дата звернення: 15.02.20)

15. Сторінка з веб-сайту Сайд приклад проведення маркетингових досліджень [Електронний ресурс] Digdata 2020 - Режим доступу до ресурсу: https://digdata.com.ua/index.php/blog/ (дата звернення: 15.02.20)

16. Сторінка з веб-сайту Сайд приклад проведення маркетингових досліджень [Електронний ресурс] Socis 2020 - Режим доступу до ресурсу: http://socis.kiev.ua/ (дата звернення: 15.02.20)

17. Сайд приклад проведення маркетингових досліджень [Електронний ресурс] Google 2020 - Режим доступу до ресурсу: https://www.google.com/intl/uk_ua/forms/about/ (дата звернення: 15.02.20)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

1 Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Web-додаток розробляється для проведення маркетингових досліджень, проведенням опитування, що допоможе визначитись з цільовою аудиторією компанії, зацікавленістю товаром чи послугами методом проведення.

1.2 Мета створення web-додатку

Метою проекту є створення web-додатку для проведення маркетингових досліджень методом опитування, який допоможе проведенню опитувань клієнтів кампанії чи підприємства.

2 Вимоги до web-додатку в цілому

2.1 Вимоги до структури й функціонування web-додатку

Розробка повинна реалізована у вигляді web-додатку, розміщеного в мережі Інтернет. Додаток буде реалізований у вигляді web-програми з можливістю створення опитувань та перегляду результатів опитування.

2.2 Вимоги до персоналу

Для роботи з додатком та підтримки додатку під персоналу не потрібно вимагатися спеціальних навичок чи умінь володіння мовами програмування, за винятком навичок роботи з персональним комп’ютером та web-браузером.

2.3 Вимоги до стилістичного оформлення сайту

Додаток повинен бути розроблений на мові програмування JavaScript з використанням бібліотек мови програмування для розробки інтерфейсу React. Дизайн повинен бути в діловому стилі, в світлих тонах з використання бібліотеки для розробки дизайну MATERIAL-UI.

3 Основні вимоги

3.1 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської сторони повинно бути:

- операційна система MacOS, Linux, Solaris, FreeBSD, OpenBSD, Windows, webOS;

- web-браузер Chrome 60+, Safari Mac IE 11+ Android 4. 4+, Chrome для Android 4 4+ iOS Safari 7+ Microsoft Edge 12+;

- підтримка JavaScript.

3.2 Функціональні вимоги

- створення простих опитування у вигляді питань з варіантами відповідей (так або ні);
- надсилання опитування потенційній аудиторії;
- можливість бачити дату початку опитування;
- можливість бачити дату останньої відповіді;
- перегляд результатів опитування у вигляді кількостей відповідей (так або ні).

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

1 Ідентифікація ідеї проекту

Web-додаток створений для проведення маркетингових досліджень, проведенням опитування, що допоможе визначитись з цільовою аудиторією, зацікавленістю товаром чи послугами, покращити ситуацію в колективі проведенням опитувань колег (наприклад визначення рівня задоволеністю працею).

Метою дипломного проекту є створення зручного сервісу проведення маркетингових досліджень опитуванням, що допоможе проводити опитування колег або клієнтів кампанії чи підприємства.

Web-додаток повинен бути реалізований в мережі Інтернет, який повинен складатись з розділів з чітко визначеними функціями.

2 Деталізація мети методом SMART

Результати деталізації методом SMART розміщені у табл. Б.1.

Продуктом дипломного проекту є web-додаток для проведення маркетингових досліджень.

Таблиця Б.1. − Деталізація мети методом SMART

Specific (конкретна)	Створити web-додаток для проведення маркетингових опитувань.
Measurable (вимірювана)	Результатом виконання проекту буде оцінка користувачів.
Achievable (досяжна)	Створення web-додатку на мові програмування JavaScript з використанням програмної платформи Node.js, бібліотеки React для розробки інтерфейсів користувачів та із використанням MongoDB для керування даними.

Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни згідно з календарним планом.

3 Описання фази розробки IT—проекту

3.1 Планування змісту структури робіт IT—проекту (WBS)

Основним інструментом для планування змісту структури робіт служить WBS діаграма − графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. Побудуємо структуру WBS, у якій детально опишемо роботи, які потрібно виконати на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рис. Б.1.

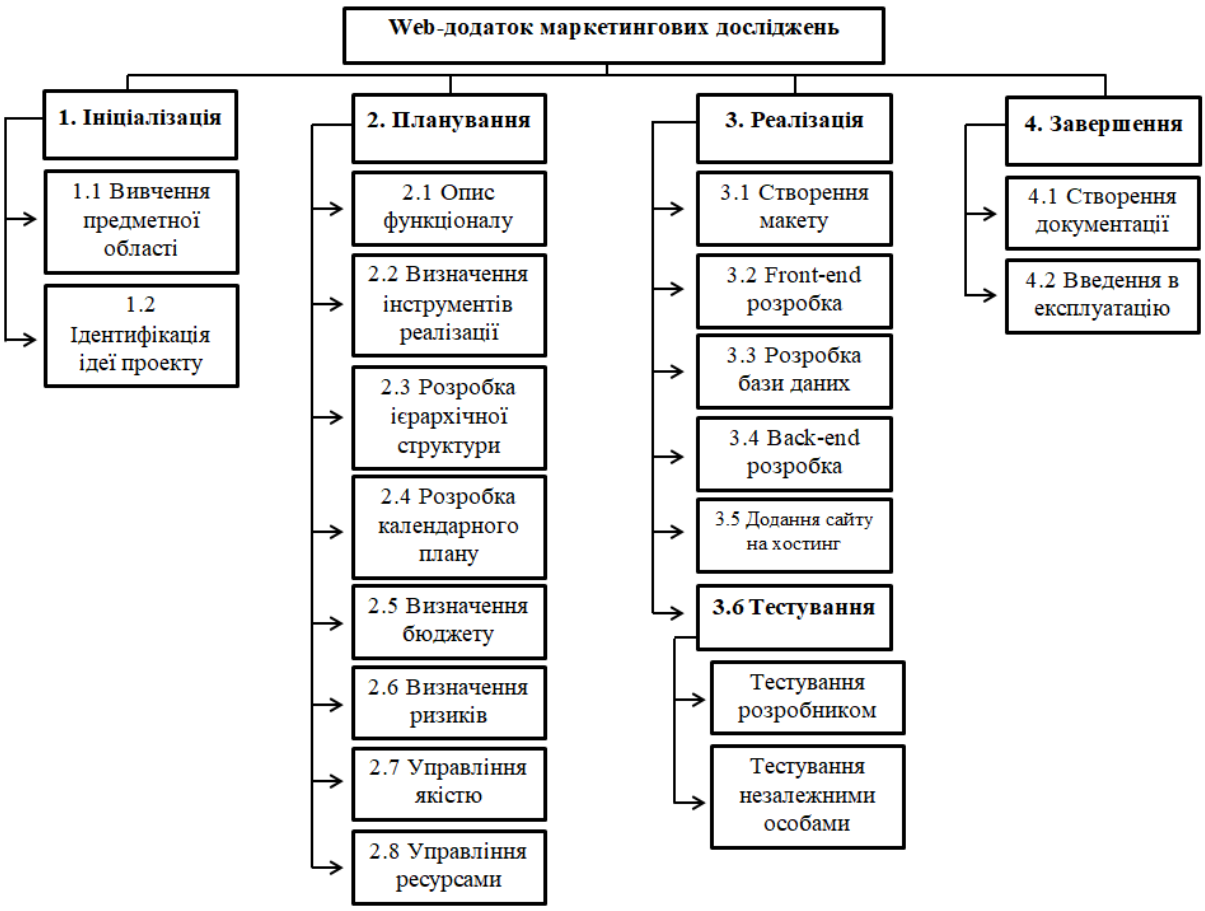


Рисунок Б.1. – *WBS. Структура робіт проекту*

3.2 Планування структури організації, для впровадження готового проекту (OBS)

Після побудови WBS розробимо організаційну структуру виконавців OBS. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Діаграма OBS зображена на рис. Б.2.

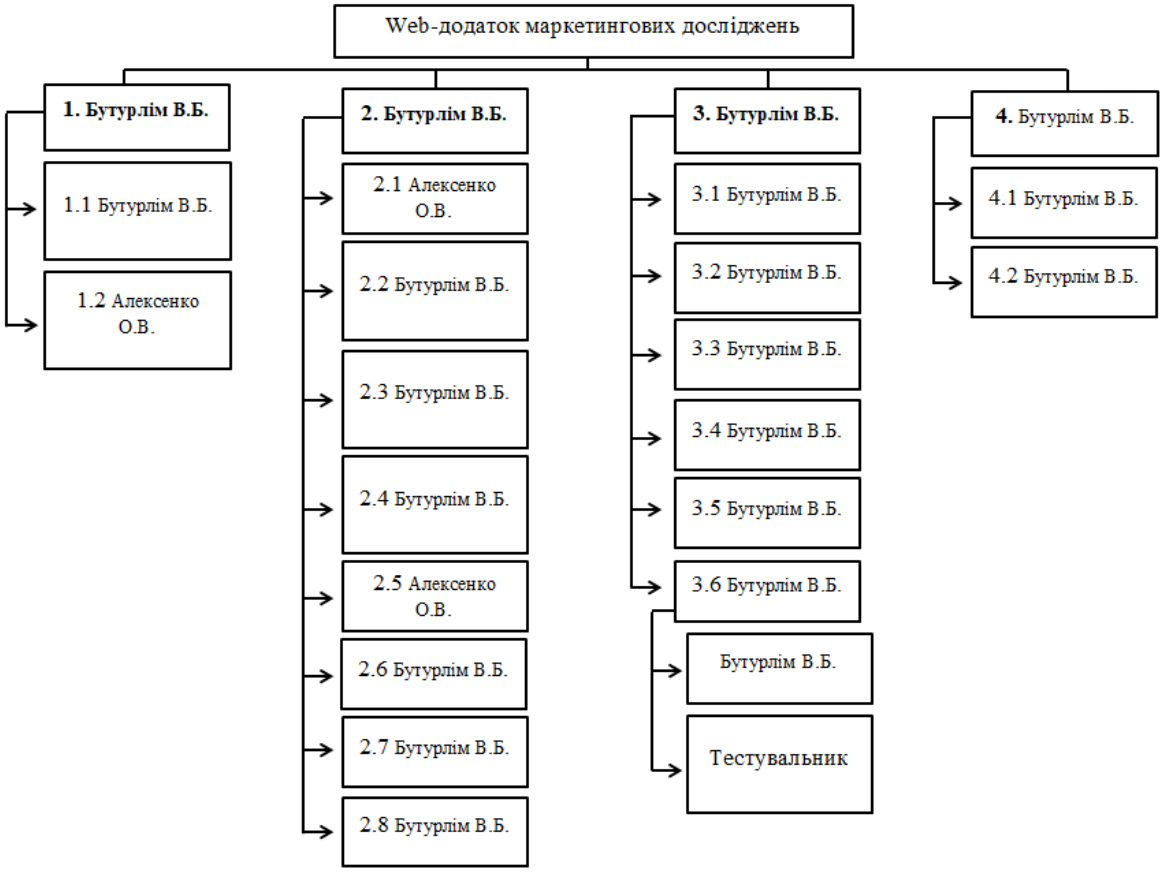


Рисунок Б.2. – *OBS. Структура робіт проекту*

3.3 Побудова матриці відповідальності (виконавців пакетів робіт)

На підставі OBS та WBS структур було побудовано матрицю відповідальності. Для кожного з виконавців була визначена його роль:

– Бутурлім В.Б. повністю відповідає за виконання задачі та приймає рішення щодо реалізації;

– Алексенко О.В. – наглядає за ходом виконання завдання, контролює процес виконання на кожному етапі розробки.

– тестувальник – проводить незалежне тестування проекту.

Матриця відповідальності представлена в табл. Б.2

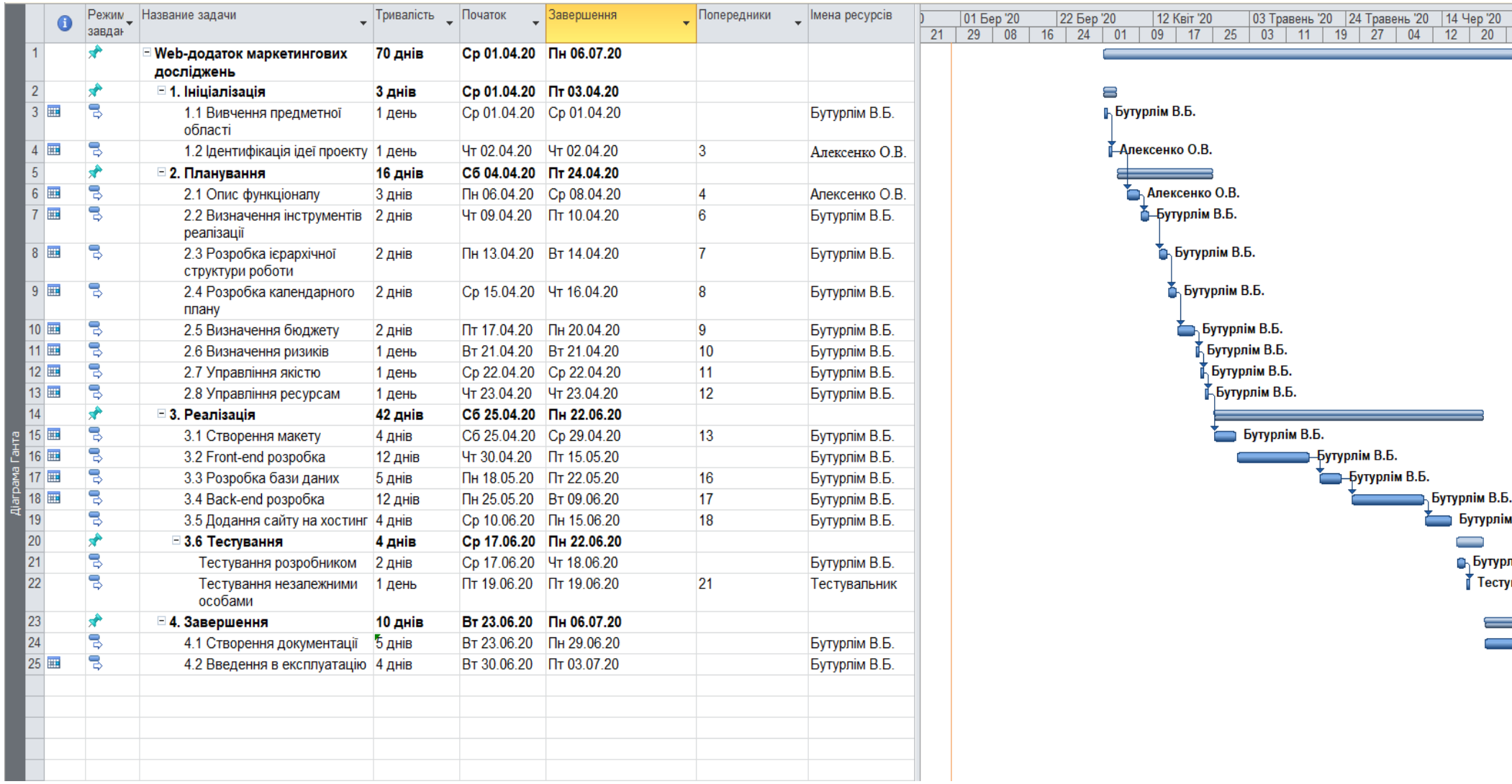
Таблиця Б.2 – Матриця відповідальності

Web-додаток маркетингових досліджень			
	Бутурлім В.Б.	АлексенкоО.В.	Тестувальник
1. Ініціалізація			
1.1 Вивчення предметної області	+		
1.2 Ідентифікація ідеї проекту		+	
2. Планування			
2.1 Опис функціоналу		+	
2.2 Визначення інструментів реалізації	+		
2.3 Розробка ієрархічної структури роботи	+		
2.4 Розробка календарного плану	+		
2.5 Визначення бюджету		+	
2.6 Визначення ризиків	+		
2.7 Управління якістю	+		
2.8 Управління ресурсам	+		
3. Реалізація			
3.1 Створення макету	+		
3.2 Front-end розробка	+		
3.3 Розробка бази даних	+		
3.4 Back-end розробка	+		
3.5 Додання сайту на хостинг	+		
3.6 Тестування			
Тестування розробником	+		
Тестування незалежними особами			+
4. Завершення			
4.1 Створення документації	+		
4.2 Введення в експлуатацію	+		

4 Побудова календарного графіку виконання ІТ—проекту (включаючи побудову часткових мережевих моделей у вигляді діаграм Ганта)

Діаграма Ганта - горизонтальна лінійна діаграма, на якій задачі проекту представляються протяжними в часі відрізками, що характеризуються датами початку та закінчення, затримками і, можливо, іншими тимчасовими параметрами.

Для побудови діаграми Ганта використовували програмний продукт MS Project рис. Б.3.



	Режим завдан	Название задачи	Тривалість	Початок	Завершення	Попередники	Імена ресурсів
1		**Web-додаток маркетингових досліджень**	**70 днів**	**Ср 01.04.20**	**Пн 06.07.20**		
2		**1. Ініціалізація**	**3 днів**	**Ср 01.04.20**	**Пт 03.04.20**		
3		1.1 Вивчення предметної області	1 день	Ср 01.04.20	Ср 01.04.20		Бутурлім В.Б.
4		1.2 Ідентифікація ідеї проекту	1 день	Чт 02.04.20	Чт 02.04.20	3	Алексенко О.В.
5		**2. Планування**	**16 днів**	**Сб 04.04.20**	**Пт 24.04.20**		
6		2.1 Опис функціоналу	3 днів	Пн 06.04.20	Ср 08.04.20	4	Алексенко О.В.
7		2.2 Визначення інструментів реалізації	2 днів	Чт 09.04.20	Пт 10.04.20	6	Бутурлім В.Б.
8		2.3 Розробка ієрархічної структури роботи	2 днів	Пн 13.04.20	Вт 14.04.20	7	Бутурлім В.Б.
9		2.4 Розробка календарного плану	2 днів	Ср 15.04.20	Чт 16.04.20	8	Бутурлім В.Б.
10		2.5 Визначення бюджету	2 днів	Пт 17.04.20	Пн 20.04.20	9	Бутурлім В.Б.
11		2.6 Визначення ризиків	1 день	Вт 21.04.20	Вт 21.04.20	10	Бутурлім В.Б.
12		2.7 Управління якістю	1 день	Ср 22.04.20	Ср 22.04.20	11	Бутурлім В.Б.
13		2.8 Управління ресурсам	1 день	Чт 23.04.20	Чт 23.04.20	12	Бутурлім В.Б.
14		**3. Реалізація**	**42 днів**	**Сб 25.04.20**	**Пн 22.06.20**		
15		3.1 Створення макету	4 днів	Сб 25.04.20	Ср 29.04.20	13	Бутурлім В.Б.
16		3.2 Front-end розробка	12 днів	Чт 30.04.20	Пт 15.05.20		Бутурлім В.Б.
17		3.3 Розробка бази даних	5 днів	Пн 18.05.20	Пт 22.05.20	16	Бутурлім В.Б.
18		3.4 Back-end розробка	12 днів	Пн 25.05.20	Вт 09.06.20	17	Бутурлім В.Б.
19		3.5 Додання сайту на хостинг	4 днів	Ср 10.06.20	Пн 15.06.20	18	Бутурлім В.Б.
20		**3.6 Тестування**	**4 днів**	**Ср 17.06.20**	**Пн 22.06.20**		
21		Тестування розробником	2 днів	Ср 17.06.20	Чт 18.06.20		Бутурлім В.Б.
22		Тестування незалежними особами	1 день	Пт 19.06.20	Пт 19.06.20	21	Тестувальник
23		**4. Завершення**	**10 днів**	**Вт 23.06.20**	**Пн 06.07.20**		
24		4.1 Створення документації	5 днів	Вт 23.06.20	Пн 29.06.20		Бутурлім В.Б.
25		4.2 Введення в експлуатацію	4 днів	Вт 30.06.20	Пт 03.07.20		Бутурлім В.Б.

Рисунок Б.3. – *Діаграма Ганта*

5 Ідентифікація ризиків

Ідентифікація ризиків – це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик. Це ітеративний процес, який періодично повторюється на всьому протязі проекту, оскільки в рамках його життєвого циклу можуть виявлятися нові ризики.

Якісний аналіз передує кількісному. Він передбачає визначення факторів ризику, ідентифікацію потенційних областей виникнення ризику, виявлення напрямків діяльності та етапів, на яких може реалізуватися ризик. Протягом якісного аналізу також встановлюється можливість кількісної оцінки ризиків, реалізація яких може вплинути на діяльність підприємства.

Кількісна оцінка ризиків часто супроводжує якісну оцінку і також вимагає процес ідентифікації ризиків. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків.

Планування реагування на ризики – це процес розробки шляхів і визначення дій із збільшення можливостей і зниження погроз для цілей проекту. Даний процес зачинається після проведення якісного і кількісного аналізу ризиків. На рисунку Б.4 розташовується класифікація ризиків.

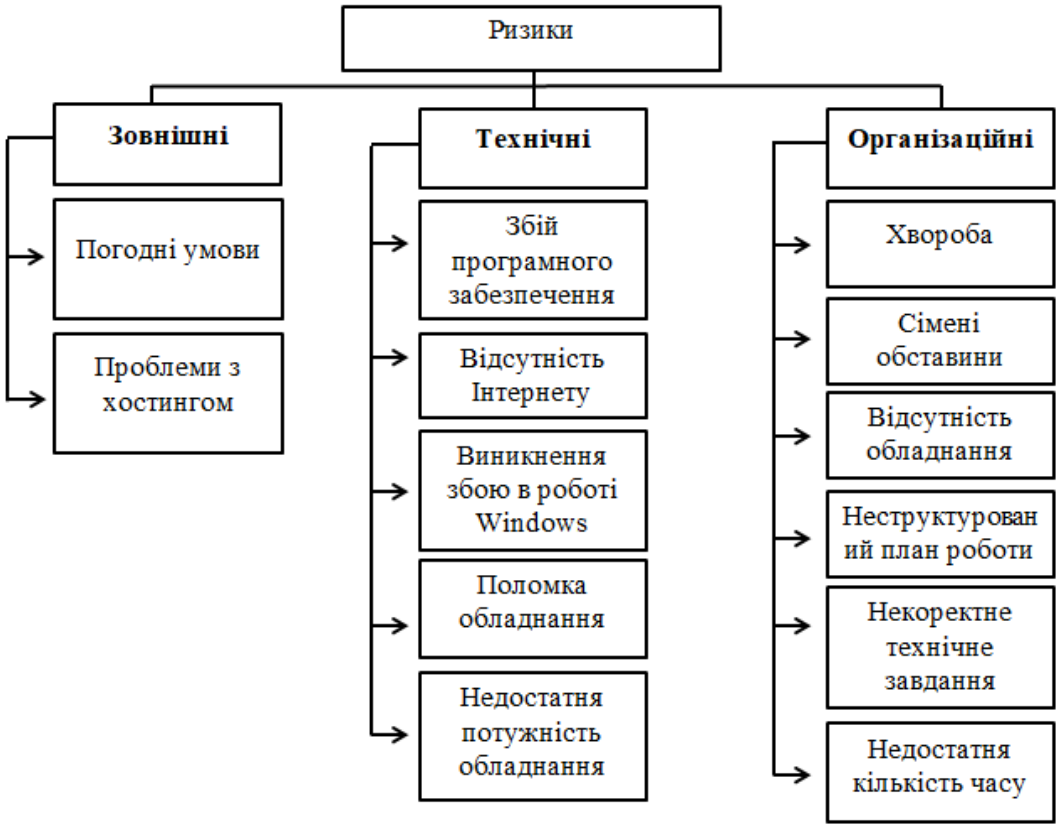


Рисунок Б.4 – *Ризики*

6.1 Матриця ризиків

Ризики представити за допомогою RBM матриці (Risk Breakdown Matrix) на рис. Б.5.

Класифікація ризиків за імовірністю виникнення:

– слабоімовірнісні;

– малоімовірнісні;

– імовірні;

– досить імовірні;

– майже імовірні.

Класифікація ризиків за імовірністю виникнення за величиною втрат:

– мінімальна;

– низька

– середня

– висока

– максимальна

Виконаємо класифікацію ризиків даного проекту. Для цього складемо табл. Б.2.

Таблиця Б.2 – Класифікація ризиків дипломного проекту

Ризик		Ймовірність виникнення	Величина втрат
Відсутність Інтернету	R1	1	3
Проблеми з хостингом	R2	1	2
Погодні умови	R3	2	2
Збій програмного забезпечення	R4	2	3
Виникнення збою в роботі Windows	R5	1	3
Хвороба	R6	2	2
Сімені обставини	R7	1	2
Недостатня потужність обладнання	R8	1	3
Неструктурований план роботи	R9	2	2
Некоректне технічне завдання	R10	2	2
Недостатня кількість часу	R11	2	2
Відсутність обладнання	R12	1	3
Поломка обладнання	R13	1	3

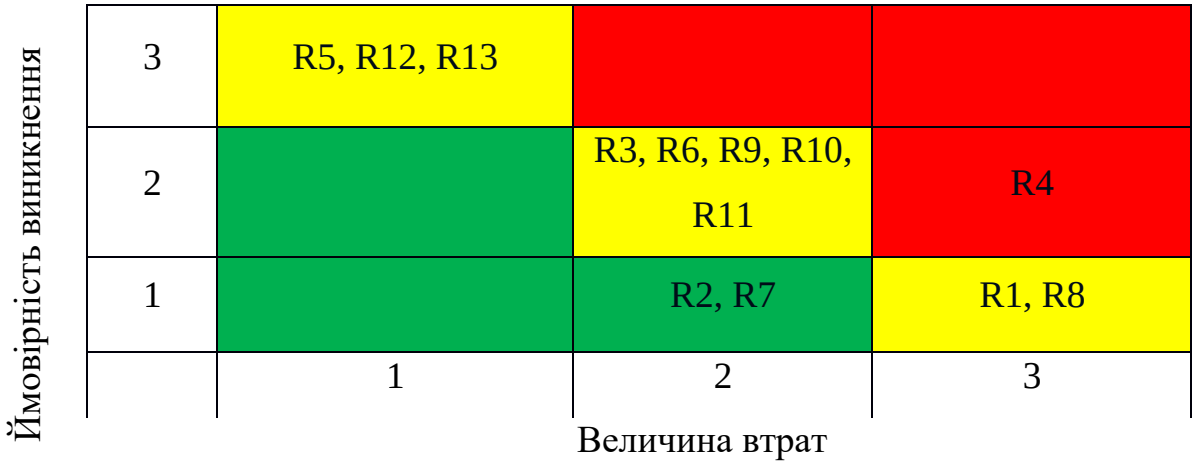


Рисунок Б.5 – *Матриця ймовірності виникнення ризиків та впливу ризику*

6.2 Класифікація за ступенем впливу та за рівнем ризику (табл. Б.3)

Класифікація за ступенем впливу:

- ігноровані (1≤R≤4);
- незначні (5≤R≤8);
- помірні (9≤R≤11);
- вагомі (12≤R≤19);
- критичні (20≤R≤25).

Класифікація за рівнем ризику:

- прийнятні ризики;
- виправданні ризики;
- недопустимі ризики;

Таблиця Б.3 – Класифікація за ступенем впливу та за рівнем ризику

Ризик		Ступінь впливу	Рівень ризику
Відсутність Інтернету	R1	18	прийнятні ризики
Проблеми з хостингом	R2	14	виправданні ризики
Погодні умови	R3	8	виправданні ризики
Збій програмного забезпечення	R4	9	недопустимі ризики
Виникнення збою в роботі Windows	R5	15	недопустимі ризики
Хвороба	R6	5	прийнятні ризики
Сімені обставини	R7	5	прийнятні ризики
Недостатня потужність обладнання	R8	8	виправданні ризики
Неструктурований план роботи	R9	8	виправданні ризики
Некоректне технічне завдання	R10	9	виправданні ризики
Недостатня кількість часу	R11	12	прийнятні ризики
Відсутність обладнання	R12	15	недопустимі ризики
Поломка обладнання	R13	18	недопустимі ризики

План по усуненню ризиків:

- використовувати потужне обладнання;

- робити резервні копії програмного продукту з хостингу;
- здійснювати проміжний контроль результатів в ході виконання проекту;
- враховувати досвід проектів-аналогів;
- проводити тестування на кожному етапі написання додатку;
- використовувати програми страхування технічних ризиків;
- ретельний вибір інструментів виконання проекту.

ДОДАТОК В

ЛІСТИНГ КОДУ WEB-ДОДАТКУ

Лістинг коду файлу package.json:

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "engines": {
    "node": "12.16.1",
    "npm": "6.13.4"
  },
  "scripts": {
    "start": "node index.js",
    "server": "nodemon index.js",
    "client": "npm run start --prefix client",
    "dev": "concurrently \"npm run server\" \"npm run client\" \"npm run webhook\"",
    "heroku-postbuild": "NPM_CONFIG_PRODUCTION=false npm install --prefix client && npm run build --prefix client",
    "webhook": "lt -p 5000 -s feedbackme77209"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "concurrently": "^5.2.0",
    "cookie-session": "^1.4.0",
    "express": "^4.17.1",
    "localtunnel": "^2.0.0",
    "lodash": "^4.17.15",
    "mongoose": "^5.9.15",
    "nodemon": "^2.0.4",
    "passport": "^0.4.1",
    "passport-google-oauth20": "^2.0.0",
```

```
    "path-parser": "^6.1.0",
    "react-redux": "^7.2.0",
    "react-router-dom": "^5.2.0",
    "redux": "^4.0.5",
    "sendgrid": "^5.2.3",
    "stripe": "^8.56.0"
  }
}
```

Лістинг коду файлу index.js:

```
const express = require('express');
const mongoose = require('mongoose');
const cookieSession = require('cookie-session');
const passport = require('passport');
const bodyParser = require('body-parser');
const keys = require('./config/keys');
require('./models/User');
require('./models/Survey');
require('./services/passport');
mongoose.connect(keys.mongoURI, { useNewUrlParser: true, useUnifiedTopology: true}, (err) => {
  if (err) {
    console.warn(err);
    return;
  }
  console.log('Database connected successfully');
})
const app = express();
app.use(bodyParser.json());
app.use(
  cookieSession({
    maxAge: 30 * 24 * 60 * 60 * 1000,
    keys: [keys.cookieKey]
  })
);
app.use(passport.initialize());
app.use(passport.session());
```

```
require('./routes/authRoutes')(app);
require('./routes/billingRoutes')(app);
require('./routes/surveyRoutes')(app);
if (process.env.NODE_ENV === 'production') {
  app.use(express.static('client/build'));
  const path = require('path');
  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
  });
}
const PORT = process.env.PORT || 5000;
app.listen(PORT);
```

Лістинг коду файлу Mailer.js:

```
const sendgrid = require('sendgrid');
const helper = sendgrid.mail;
const keys = require('../config/keys');

class Mailer extends helper.Mail {
  constructor({ subject, recipients }, content) {
    super();

    this.sgApi = sendgrid(keys.sendGridKey);
    this.from_email = new helper.Email('no-reply@emaily.com');
    this.subject = subject;
    this.body = new helper.Content('text/html', content);
    this.recipients = this.formatAddresses(recipients);

    this.addContent(this.body);
    this.addClickTracking();
    this.addRecipients();
  }

  formatAddresses(recipients) {
    return recipients.map(({ email }) => {
```

```
      return new helper.Email(email);
    });
  }

  addClickTracking() {
    const trackingSettings = new helper.TrackingSettings();
    const clickTracking = new helper.ClickTracking(true, true);

    trackingSettings.setClickTracking(clickTracking);
    this.addTrackingSettings(trackingSettings);
  }

  addRecipients() {
    const personalize = new helper.Personalization();

    this.recipients.forEach(recipient => {
      personalize.addTo(recipient);
    });
    this.addPersonalization(personalize);
  }

  async send() {
    const request = this.sgApi.emptyRequest({
      method: 'POST',
      path: '/v3/mail/send',
      body: this.toJSON()
    });

    const response = await this.sgApi.API(request);
    return response;
  }
}

module.exports = Mailer;
```

Лістинг коду файлу passport.js:

```
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const mongoose = require('mongoose');
const keys = require('../config/keys');

const User = mongoose.model('users');

passport.serializeUser((user, done) => {
  done(null, user.id);
});

passport.deserializeUser((id, done) => {
  User.findById(id).then(user => {
    done(null, user);
  });
});

passport.use(
  new GoogleStrategy(
    {
      clientID: keys.googleClientID,
      clientSecret: keys.googleClientSecret,
      callbackURL: '/auth/google/callback',
      proxy: true
    },
    async (accessToken, refreshToken, profile, done) => {
      const existingUser = await User.findOne({ googleId: profile.id });

      if (existingUser) {
        return done(null, existingUser);
      }

      const user = await new User({ googleId: profile.id }).save();
      done(null, user);
    }
```

```
  )
);
```

Лістинг коду файлу surveyTemplate.js:

```
const keys = require('../../config/keys');

module.exports = survey => {
  return `
    <html>
      <body>
        <div style="text-align: center;">
          <h3>Зробіть свій вибір!</h3>
          <p>Будь ласка, дайте відповідь на наступне запитання:</p>
          <p>${survey.body}</p>
          <div>
            <a href="${keys.redirectDomain}/api/surveys/${survey.id}/yes">Так</a>
          </div>
          <div>
            <a href="${keys.redirectDomain}/api/surveys/${survey.id}/no">Ні</a>
          </div>
        </div>
      </body>
    </html>
  `;
};
```

Лістинг коду файлу authRoutes.js:

```
const passport = require('passport');


module.exports = app => {
  app.get(
    '/auth/google',
    passport.authenticate('google', {
      scope: ['profile', 'email']
    })
```

```
  );

  app.get(
    '/auth/google/callback',
    passport.authenticate('google'),
    (req, res) => {
      res.redirect('/surveys');
    }
  );

  app.get('/api/logout', (req, res) => {
    req.logout();
    res.redirect('/');
  });

  app.get('/api/current_user', (req, res) => {
    res.send(req.user);
  });
};
```

Лістинг коду файлу billingRoutes.js:

```
const keys = require('../config/keys');
const stripe = require('stripe')(keys.stripeSecretKey);
const requireLogin = require('../middlewares/requireLogin');

module.exports = app => {
  app.post('/api/stripe', requireLogin, async (req, res) => {
    const charge = await stripe.charges.create({
      amount: 500,
      currency: 'usd',
      description: '$5 за 5 кредитів',
      source: req.body.id
    });

    req.user.credits += 5;
    const user = await req.user.save();
```

```
    res.send(user);
  });
};
```

Лістинг коду файлу surveyRoutes.js:

```
const _ = require('lodash');
const { Path } = require('path-parser');
const { URL } = require('url');
const mongoose = require('mongoose');
const requireLogin = require('../middlewares/requireLogin');
const requireCredits = require('../middlewares/requireCredits');
const Mailer = require('../services/Mailer');
const surveyTemplate = require('../services/emailTemplates/surveyTemplate');

const Survey = mongoose.model('surveys');

module.exports = app => {

    app.delete('/api/surveys/', requireLogin, async (req, res) => {
        try {
            await Survey.deleteOne({ _id: req.body.surveyId });
            res.send({});
        } catch (err) {
            res.status(501).send(err);
        }
    })

    app.get('/api/surveys', requireLogin, async (req, res) => {
        const surveys = await Survey.find({ _user: req.user.id })
            .select({ recipients: false }) // исключить из результата recipients

        res.send(surveys);
    });
    app.get('/api/surveys/:surveyId/:choice', (req, res) => {
        res.send('Дякую за ваш голос!');
```

```
  })
  app.post('/api/surveys/webhooks', (req, res) => {
    const p = new Path('/api/surveys/:surveyId/:choice');
    _.chain(req.body)
      .map(({ email, url, event }) => {
        const match = p.test(new URL(url).pathname);
        if (match && event === 'click') {
          return { email, surveyId: match.surveyId, choice: match.choice };
        }
      })
      .each(event => {
        Survey.updateOne({
          _id: event.surveyId, // найти survey с id == surveyId, который имеет recipients:
        }, {
            $inc: { [event.choice]: 1 },
($inc) его на 1;
            $set: { 'recipients.$.responded': true },
            lastResponded: new Date()
          }).exec();
      })
      .value();
    res.send({ message: 'OK' });
  });

  app.post('/api/surveys', requireLogin, requireCredits, async (req, res) => {
    const { title, subject, body, recipients } = req.body;

    const survey = new Survey({
      title,
      subject,
      body,
      recipients: recipients.split(',').map(email => ({ email: email.trim() })),
      //refactored from:
      //recipients: recipients.split(',').map(recipient => { return { email: recipient } })
      _user: req.user.id,
```

```
        dateSent: Date.now()
      });

      const mailer = new Mailer(survey, surveyTemplate(survey));
      try {
        await mailer.send();
        await survey.save();
        req.user.credits -= 1;
        const user = await req.user.save();

        res.send(user);
      } catch (err) {
        res.status(422).send(err);
      }


  });
};
```

Лістинг коду файлу Recipient.js:

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const recipientSchema = new Schema({
 email: String,
 responded: { type: Boolean, default: false }
});
module.exports = recipientSchema;
```

Лістинг коду файлу Survey.js:

```
const mongoose = require('mongoose');
const { Schema } = mongoose;
const RecipientSchema = require('./Recipient');


const surveySchema = new Schema({
 title: String,
```

```
  body: String,
  subject: String,
  recipients: [RecipientSchema],
  yes: { type: Number, default: 0 },
  no: { type: Number, default: 0 },
  _user: { type: Schema.Types.ObjectId, ref: 'User' },
  dateSent: Date,
  lastResponded: Date
});


mongoose.model('surveys', surveySchema);
```

Лістинг коду файлу requireCredits.js:

```
module.exports = (req, res, next) => {
  if (req.user.credits < 1) {
    return res.status(403).send({ error: 'Не вистачає кредитів!' });
  }

  next();
};
```

Лістинг коду файлу requireLogin.js:

```
module.exports = (req, res, next) => {
  if (!req.user) {
    return res.status(401).send({ error: 'Ви повинні увійти!' });
  }

  next();
};
```

Лістинг коду файлу prod.js:

```
module.exports = {
    googleClientID: process.env.GOOGLE_CLIENT_ID,
    googleClientSecret: process.env.GOOGLE_CLIENT_SECRET,
    mongoURI: process.env.MONGO_URI,
    cookieKey: process.env.COOKIE_KEY,
    stripePublishableKey: process.env.STRIPE_PUBLISHABLE_KEY,
```

```
  stripeSecretKey: process.env.STRIPE_SECRET_KEY,
  sendGridKey: process.env.SEND_GRID_KEY,
  redirectDomain: process.env.REDIRECT_DOMAIN
};
```

Лістинг коду файлу index.js:

```
import 'materialize-css/dist/css/materialize.min.css';
import React from 'react';
import ReactDOM from 'react-dom';
import { Provider } from 'react-redux';
import { createStore, applyMiddleware } from 'redux';
import reduxThunk from 'redux-thunk';

import App from './components/App';
import reducers from './reducers';

import axios from 'axios';
window.axios = axios;

const store = createStore(reducers, {}, applyMiddleware(reduxThunk));

ReactDOM.render(
  <Provider store={store}><App /></Provider>,
  document.querySelector('#root')
);
```

Лістинг коду файлу setupProxy.js:

```
const { createProxyMiddleware } = require('http-proxy-middleware');
const proxy = require('http-proxy-middleware');
module.exports = function(app) {
  app.use(createProxyMiddleware('/auth/google',
    { target: 'http://localhost:5000' }
  ));
  app.use(createProxyMiddleware('/api/*',
  { target: 'http://localhost:5000' }
));
}
```

Лістинг коду файлу validateEmails.js:

```
const re = /^[a-zA-Z0-9.!#$%&’*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;

export default emails => {
  const invalidEmails = emails
    .split(',')
    .map(email => email.trim())
    .filter(email => re.test(email) === false);

  if (invalidEmails.length) {
    return `Недійсна електронна пошта: ${invalidEmails}`;
  }

  return;
};
```

Лістинг коду файлу authReducer.js:

```
import { FETCH_USER } from '../actions/types';


export default function(state = null, action) {
  switch (action.type) {
    case FETCH_USER:
      return action.payload || false;
    default:
      return state;
  }
}
```

Лістинг коду файлу index.js:

```
import { combineReducers } from 'redux';
import { reducer as reduxForm } from 'redux-form';
import authReducer from './authReducer';
import surveysReducer from './surveysReducer';


export default combineReducers({
```

```
  auth: authReducer,
  form: reduxForm,
  surveys: surveysReducer
});
```

Лістинг коду файлу surveysReducer.js:

```
import { FETCH_SURVEYS } from '../actions/types';


export default function(state = [], action) {
  switch (action.type) {
    case FETCH_SURVEYS:
      return action.payload;
    default:
      return state;
  }
}
```

Лістинг коду файлу App.js:

```
import React, { Component } from 'react';
import { BrowserRouter, Route } from 'react-router-dom';
import { connect } from 'react-redux';
import * as actions from '../actions';
import Header from './Header';
import Landing from './Landing';
import Dashboard from './Dashboard';
import SurveyNew from './surveys/SurveyNew';

class App extends Component {
  componentDidMount() {
    this.props.fetchUser();
  }

  render() {
    return (
      <div className="container">
        <BrowserRouter>
```

```
          <div>
            <Header />
            <Route exact path="/" component={Landing} />
            <Route exact path="/surveys" component={Dashboard} />
            <Route path="/surveys/new" component={SurveyNew} />
          </div>
        </BrowserRouter>
      </div>
    );
  }
}

export default connect(null, actions)(App);
```

Лістинг коду файлу Dashboard.js:

```
import React from 'react';
import { Link } from 'react-router-dom';
import SurveyList from './surveys/SurveyList';


const Dashboard = () => {
  return (
    <div>
      <SurveyList />
      <div className="fixed-action-btn">
        <Link to="/surveys/new" className="btn-floating btn-large red">
          <i className="material-icons">add</i>
        </Link>
      </div>
    </div>
  );
};


export default Dashboard;
```

Лістинг коду файлу Header.js:

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { Link } from 'react-router-dom';
import Payments from './Payments';

class Header extends Component {
 renderContent() {
  switch (this.props.auth) {
   case null:
    return;
   case false:
    return <li><a href="/auth/google">Вхід із Google</a></li>;
   default:
    return [
     <li key="1"><Payments /></li>,
     <li key="3" style={{ margin: '0 10px' }}>
      Кредити: {this.props.auth.credits}
     </li>,
     <li key="2"><a href="/api/logout">Вийти</a></li>
    ];
  }
 }

 render() {
  return (
   <nav style={{ backgroundColor: '#6eb9ee' }}>
    <div className="nav-wrapper">
     <Link
      to={this.props.auth ? '/surveys' : '/'}
      className="left brand-logo"
     >
      Опитування
     </Link>
     <ul className="right">
      {this.renderContent()}
```

```
          </ul>
        </div>
      </nav>
    );
  }
}

function mapStateToProps({ auth }) {
  return { auth };
}

export default connect(mapStateToProps)(Header);
```

Лістинг коду файлу Landing.js:

```
import React from 'react';

const Landing = () => {
  return (
    <div style={{ textAlign: 'center' }}>
      <h1>
      Опитування!
      </h1>
      Отримайте відгуки від своїх користувачів.
    </div>
  );
};

export default Landing;
```

Лістинг коду файлу Payments.js:

```
import React, { Component } from 'react';
import StripeCheckout from 'react-stripe-checkout';
import { connect } from 'react-redux';
import * as actions from '../actions';

class Payments extends Component {
  render() {
```

```
    return (
      <StripeCheckout
        name="Опитування"
        description="$5 за 5 кредитів"
        amount={500}
        token={token => this.props.handleToken(token)}
        stripeKey={process.env.REACT_APP_STRIPE_KEY}
      >
        <button className="btn">
        Додати кредити
        </button>
      </StripeCheckout>
    );
  }
}

export default connect(null, actions)(Payments);
```

Лістинг коду файлу formFields.js:

```
export default [
  { label: 'Назва кампанії', name: 'title' },
  { label: 'Тема', name: 'subject' },
  { label: 'Текст/ питання', name: 'body' },
  { label: 'Список одержувачів', name: 'recipients' }
];
```

Лістинг коду файлу SurveyField.js:

```
import React from 'react';

export default ({ input, label, meta: { error, touched } }) => {
  return (
    <div>
      <label>{label}</label>
      <input {...input} style={{ marginBottom: '5px' }} />
      <div className="red-text" style={{ marginBottom: '20px' }}>
        {touched && error}
```

```
    </div>
   </div>
  );
};
```

Лістинг коду файлу SurveyForm.js:

```
import _ from 'lodash';
import React, { Component } from 'react';
import { reduxForm, Field } from 'redux-form';
import { Link } from 'react-router-dom';
import SurveyField from './SurveyField';
import validateEmails from '../../utils/validateEmails';
import formFields from './formFields';

class SurveyForm extends Component {
  renderFields() {
    return _.map(formFields, ({ label, name }) => {
      return (
        <Field
          key={name}
          component={SurveyField}
          type="text"
          label={label}
          name={name}
        />
      );
    });
  }

  render() {
    return (
      <div>
        <form onSubmit={this.props.handleSubmit(this.props.onSurveySubmit)}>
          {this.renderFields()}
          <Link to="/surveys" className="red btn-flat white-text">
          Скасувати
```

```
          </Link>
          <button type="submit" className="teal btn-flat right white-text">
          Далі
            <i className="material-icons right">done</i>
          </button>
        </form>
      </div>
    );
  }
}

function validate(values) {
  const errors = {};

  errors.recipients = validateEmails(values.recipients || '');

  _.each(formFields, ({ name }) => {
    if (!values[name]) {
      errors[name] = 'Ви повинні ввести дані';
    }
  });

  return errors;
}

export default reduxForm({
  validate,
  form: 'surveyForm',
  destroyOnUnmount: false
})(SurveyForm);
```

Лістинг коду файлу SurveyFormReview.js:

```
import _ from 'lodash';
import React from 'react';
import { connect } from 'react-redux';
import formFields from './formFields';
```

```
import { withRouter } from 'react-router-dom';
import * as actions from '../../actions';

const SurveyFormReview = ({ onCancel, formValues, submitSurvey, history }) => {
  const reviewFields = _.map(formFields, ({ name, label }) => {
    return (
      <div key={name}>
        <label>{label}</label>
        <div>
          {formValues[name]}
        </div>
      </div>
    );
  });

  return (
    <div>
      <h5>Все правильно?</h5>
      {reviewFields}
      <button
        className="yellow darken-3 white-text btn-flat"
        onClick={onCancel}
      >
        Назад
      </button>
      <button
        onClick={() => submitSurvey(formValues, history)}
        className="green btn-flat right white-text"
      >
        Надіслати опитування
        <i className="material-icons right">email</i>
      </button>
    </div>
  );
};
```

```
function mapStateToProps(state) {
  return { formValues: state.form.surveyForm.values };
}

export default connect(mapStateToProps, actions)(withRouter(SurveyFormReview));
```

Лістинг коду файлу SurveyList.js:

```
import React from 'react';
import { connect } from 'react-redux';
import { fetchSurveys, deleteSurvey } from '../../actions';

class SurveyList extends React.Component {

   componentDidMount() {
      this.props.fetchSurveys();
   }

   renderSurveys() {
      return this.props.surveys.reverse().map(survey => {
         return (
            <div className="card darken-1" key={survey._id}>
               <div className="card-content">
                  <span className="card-title">{survey.title}</span>
                  <p>{survey.body}</p>
                  <p className="right">Дата відправлення: {new
Date(survey.dateSent).toLocaleDateString()}</p>
                  <br />
                  <p className="right">Остання відповідь:
                     {survey.lastResponded ? new Date(survey.lastResponded).toLocaleDateString() :
'Немає відповідей'}
                  </p>
               </div>
               <div className="card-action">
                  <span style={{ marginRight: '20px', color: 'green' }}>Так: {survey.yes}</span>
                  <span style={{ color: 'red' }}>Hi: {survey.no}</span>
```

```
                <button
                  className="red right btn-flat white-text"
                  style={{ marginTop: '-5px' }}
                  onClick={() => { this.props.deleteSurvey(survey._id) }}
                >Видалити</button>
              </div>
            </div>
          );
        })
    }

    render() {
      return (
        <div>{this.renderSurveys()}</div>
      );
    }
}

function mapStateToProps({ surveys }) {
    return { surveys };
}

export default connect(mapStateToProps, { fetchSurveys, deleteSurvey })(SurveyList);
```

Лістинг коду файлу SurveyNew.js:

```
import React, { Component } from 'react';
import { reduxForm } from 'redux-form';
import SurveyForm from './SurveyForm';
import SurveyFormReview from './SurveyFormReview';

class SurveyNew extends Component {
  state = { showFormReview: false };

  renderContent() {
    if (this.state.showFormReview) {
      return (
```

```
        <SurveyFormReview
          onCancel={() => this.setState({ showFormReview: false })}
        />
      );
    }

    return (
      <SurveyForm
        onSurveySubmit={() => this.setState({ showFormReview: true })}
      />
    );
  }

  render() {
    return (
      <div>
        {this.renderContent()}
      </div>
    );
  }
}

export default reduxForm({
  form: 'surveyForm'
})(SurveyNew);
```