Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

WEB-ЗАСТОСУНОК ДЛЯ ПРОХОДЖЕННЯ ОПИТУВАНЬ. ЧАСТИНА 2. КЛІЄНТСЬКА ЧАСТИНА

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Ганевич В. М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Колодний В.В.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., доцент каф. АІІТ
Барабан М. В.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«07» 03 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Ганевичу Владиславу Миколайовичу

1. Тема роботи: WEB-застосунок для проходження опитувань. Частина 2. Клієнтська частина.
керівник роботи: Колодний Володимир Володимирович, к.т.н., доц.
затверджені наказом вищого навчального закладу «11» 03 2024 року №90

2. Термін подання студентом роботи 27 05 2024р.

3. Вихідні дані до роботи:
Кількість варіантів аутентифікації – 2;
Кількість типів запитань – 3;
Кількість можливих питань – 7-10;
Кількість аналізованих емоцій – 1-9;

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; обгрунтування доцільності розробки WEB-застосунку для проходження опитувань; проектування WEB-застосунку для проходження опитувань; програмна реалізація WEB-застосунку для проходження опитувань; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
схема алгоритму роботи WEB-застосунку для проходження опитувань; загальна структурна схема функціонування WEB-застосунку для проходження опитувань; структура принципу роботи WEB-застосунку для проходження опитувань; загальна UML-діаграма класів; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Колодний В.В., к.т.н., доц. каф. КН		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Обґрунтування доцільності розробки WEB-застосунку для проходження опитувань	06.05.2024	11.05.2024	
2	Проектування WEB-застосунку для проходження опитувань	12.05.2024	14.05.2024	
3	Програмна реалізація WEB-застосунку для проходження опитувань	15.05.2024	19.05.2024	
4	Розробка інструкції користувача	20.05.2024	26.05.2024	
5	Висновки та аналіз отриманих результатів	27.05.2024	02.06.2024	
6	Оформлення матеріалів до захисту БДР	03.06.2024	06.06.2024	

Студент _____ **Ганевич В.М.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Колодний В.В**
(підпис) (прізвище та ініціали

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 102 сторінок формату А4, на яких є 18 рисунків, 5 таблиць, список використаних джерел містить 20 найменувань.

Метою роботи є розширення функціональних можливостей WEB-застосунку для проходження опитувань.

У загальній частині роботи на основі аналізу існуючих технічних рішень, методів та технологій доведена доцільність розробки WEB-застосунку для проходження опитувань.

Для WEB-застосунку розроблено алгоритм роботи його програмного модуля. Програмний продукт розроблений в інтегрованому середовищі Visual Studio Code за допомогою мови програмування JavaScript, HTML, CSS та фреймворку React.

Розроблено програмний продукт та проведено його тестування. Результати тестування розробки вказують на те, що основні функції WEB-застосунку для проходження опитувань реалізовано.

Ключові слова: проходження опитувань, обробка інформації, WEB-застосунок.

ABSTRACT

The bachelor thesis consists of 102 pages of A4 format, on which contain 18 figures, 5 tables, and a list of references contains 20 titles.

The method of operation is to expand the functionality of the WEB application for passing surveys.

In the general part of the work, based on the analysis of existing technical solutions, methods and technologies, the expediency of developing a WEB application for passing surveys is proven.

The algorithm of its software module was developed for the WEB application. The software product is developed in the integrated Visual Studio Code environment using JavaScript, HTML, CSS and the React framework.

A software product was developed and tested. The results of development testing indicate that the main functions of the WEB application for passing surveys have been implemented.

Keywords: passing surveys, information processing, WEB-application.

ЗМІСТ

ВСТУП

Актуальність теми. У нас час існує велика кількість людей, у кожного з яких присутня своя думка про певну подію, явище, ситуацію, тому для збору і оцінки думок людей придумали онлайн-опитування, які допомагають систематизувати ці дані з використанням платформ для проведення онлайн-опитувань.

З розвитком WEB-технологій, створення платформ для проведення опитувань стало доволі актуальним. Вони дозволяють швидко та ефективно збирати великі обсяги даних від респондентів, які можуть перебувати у будь-якому куточку світу. Такий підхід забезпечує широкий діапазон охоплення аудиторії, підвищує достовірність та актуальність зібраної інформації.

Сьогодні, основним інструментом для проходження онлайн-опитувань є WEB-застосунки, скористатись якими з легкістю можна за допомогою ПК чи смартфону. Маючи доступ до Інтернету та мінімальну кількість часу, кожен може висловити свою думку стосовно певного питання у форматі відповіді на опитування, скориставшись спеціалізованим WEB-застосунком. Саме тому, WEB-застосунки для проходження опитувань у наш час стають невід'ємною частиною психологічних, наукових, демографічних та інших досліджень, маркетингових кампаній, соціальних опитувань та багатьох інших сфер діяльності.

Отже, розробка WEB-застосунку для проходження опитувань має практичне значення.

Метою дослідження є розширення функціональних можливостей WEB-застосунку для проходження опитувань. Основна особливість розробки даного WEB-застосунку являє собою можливість відповідати на питання опитувань у вигляді спектру емоцій, що дозволить розрізнити ставлення респондента до тематики опитувань у вигляді деякого емоційного портрету.

Об’єктом дослідження є процес формування статистичних даних опитувань на основі відповідей респондентів.

Предметом дослідження є програмні засоби для реалізації та платформи для розробки WEB-застосунку для проходження опитувань.

Задачі дослідження:

1. Обґрунтувати доцільність розробки WEB-застосунку для проходження опитувань;

2. Спроектувати WEB-застосунок для проходження опитувань;

3. Програмно реалізувати WEB-застосунок для проходження опитувань;

4. Виконати тестування програми та провести аналіз отриманих результатів.

5. Розробити інструкцію користувача.

Апробація роботи.

Результати досліджень було апробовано на LIII науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2024) м. Вінниці у 2024 р. [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-ЗАСТОСУНКУ ДЛЯ ПРОХОДЖЕННЯ ОПИТУВАНЬ

Проведення опитувань завжди було важливою частиною соціального життя людей, для прикладу, можна навести президентські вибори, під час яких усе доросле населення країни заповнює анкети-бланки для вибору наступного президента, проте це доволі довготривалий і трудомісткий процес, адже кількість бланків опитування досягає мільйонів одиниць, на перевірку яких потрібно дуже багато часу. Для спрощення подібного процесу були придумані онлайн-опитування.

Онлайн-опитування – це метод збору соціологічної інформації про досліджуваний об'єкт під час анкетування з використанням веб-ресурсів. Опитування бувають соціологічні, політологічні, маркетингові, психологічні — залежно від предмету дослідження. Залежно від кількості опитуваних (вибірки, вибіркової сукупності) вони також можуть бути масовими, вибірковими, індивідуальними, експертними. Також використовується для вимірювання «громадської думки» з різних питань [2].

Онлайн-опитування, як метод збору інформації, має безліч переваг, які роблять його привабливим для використання в різних сферах. Завдяки використанню WEB-ресурсів опитування можна розпочати миттєво, без необхідності друкування анкет або організації територіальної доставки. Крім того, результати опитувань можуть бути отримані швидко, нерідко вже через кілька годин після їх початку, що дозволяє оперативно аналізувати зібрані дані та приймати відповідні рішення.

Онлайн-опитування мають свої переваги та недоліки, серед яких можна виокремити наступні:

Переваги:

1. Глобальна доступність у будь-якій точці світу;
2. Значна економія часу у порівнянні з анкетувальними опитуваннями;
3. Висока гнучкість при створенні і проходженні опитувань;

4. Кросплатформність, за допомогою якої можна проходити опитування на різних пристроях.

Недоліки:

1. Обмеженість проходження опитувань при відсутності Інтернет-з’єднання;

2. Надійність відповідей. Респонденти можуть спеціально вказувати недостовірні відповіді.

3. Обмеженість варіації відповідей на питання опитувань.

Загалом, звітування та збір даних шляхом проведення опитувань є важливою складовою багатьох діяльностей у сучасному світі, будь то дослідження, аналіз ринку, оцінка задоволеності клієнтів або просто збір зворотного зв'язку. У зв'язку з розвитком технологій і Інтернету велика частина цього процесу переноситься в онлайн-середовище.

1.1 Огляд середовищ для реалізації програмного продукту

З розвитком сучасних технолоній, WEB-застосунки для проходження опитувань стали зручною альтернативою застарілим опитуванням, які раніше проводили у формі паперових чи усних анкетувань, що потребували доволі велику кількість часу та були доволі трудомістким процесом. Такі застосунки дозволяють збирати дані від великої кількості респондентів у режимі реального часу без прикладання вагомих зусиль. Зараз, майже кожна людина може створити чи долучитися до опитування всього у декілька кліків мишкою, маючи комп’ютерний пристрій та підключення до Інтернету. Проте, розробникам, для створення ефективного WEB-застосунку для проходження опитувань, необхідно обрати середовище для розробки та розгортання застосунку [3].

Ключовим фактором вибору середовиша розробки і розгортання застосунку для проведення опитувань є кросплатформність. Платформами можуть слугувати ПК, смартфони та WEB-ресурси. Кросплатформність дозволяє забезпечити доступ до застосунку користувачам різних платформ, що є великою перевагою і дозволяє користуватись ним з різних пристроїв, операційних систем

та браузерів. В таблиці 1.1 наведена порівняльна характеристика платформ для розробки та розгортання WEB-застосунку для проходження опитувань.

Таблиця 1.1 – Порівняльна характеристика платформ для розробки та розгортання WEB-застосунку для проходження опитувань

Аспекти	ПК	Смартфон	WEB-ресурс
Доступність	Зручний доступ вдома	Мобільний доступ у будь-якому місці	Зручний доступ на пристроях з доступом до Інтернету
Функціональність	Великі можливості, дозволяє використовувати спеціалізовані побічні засоби	Обмежений функціонал відносно ПК, але це компенсується мобільністю	Відносно обмежений функціонал порівняно з ПК чи смартфоном, проте з універсальним доступу
Операційні системи	Windows, macOS, Linux	Android, iOS	Сумісний з усіма сучасними браузерами
Масштабування	Низький рівень масштабування, якщо ПК стаціонарний	Середній рівень масштабування, підходить для відносно невеликих груп осіб	Підтримка різних масштабів аудиторії, але з обмеженими функціями
Встановлення	Потребує встановлення окремого додатку	Потребує встановлення окремого додатку	Не потребує установки, доступ через веб-браузер
Зручність використання	Зручність у використанні вдома чи в офісі	Мобільність, компактність	Універсальність доступу з будь яких пристроїв
Безпека	Високий ступінь захищеності даних, проте існує загроза вірусів	Середній ступінь захищеності даних через можливість втрати пристрою	Ступінь захищеності даних залежить від браузера, який використовується

Аналізуючи таблицю 1.1, в якій порівняно платформи для розробки та розгортання WEB-застосунку для проходження опитувань, можна дійти висновку, що найбільш вдалою платформою для розробки та розгортання застосунку найкраще підходить WEB-ресурс.

Основні переваги для розгортання застосунку саме на WEB-ресурсі полягають у їх компактності – розробнику не потрібно створювати спеціальну клієнтську версію додатку, який потрібно було б завантажувати користувачеві, в даному випадку програма буде подана у вигляді веб-сторінок у браузерах; можливість відкривати застосунок у різних браузерах, без значної зміни програмного коду; знижене споживання ресурсів пристрою, на відміну від додатків, які вимагали б значно більшу кількість їх кількість.

Однак, розгортання WEB-застосунку на WEB-ресурсі може мати певні обмеження функціоналу, на відміну від спеціалізованих додатків, призначених для ПК чи смартфонів. Наприклад, застосунок може бути дещо функціонально гірший через особливості реалізації певних його компонентів, без використання певних технологій, які доступні безпосередньо на пристроях, також доступність певних функцій може прямо залежати від браузера чи обмеженості платформи.

Загалом, WEB-ресурс для розробки та розгортання WEB-застосунку для проходження опитувань є доволі зручним та універсальним варіантом, адже він дозволяє користуватися застосунком з будь-якого пристрою, будь то ПК чи смартфон, на різних платформах та операційних системах, при чому потребує мінімального доопрацювання коду для адаптації під різні браузери на різних платформах, без створення різних версій додатків, як це було б потрібно, наприклад, для ПК та смартфонів.

1.2 Огляд систем-аналогів

Серед найвідоміших аналогів WEB-застосунків для проходження опитувань можна виокремити наступні: Google Forms, SurveyMonkey, Typeform та Qualtrics.

Google Forms (рис. 1.1) підходить для простих опитувань і анкетувань, особливо для невеликих проектів, освітніх цілей та внутрішніх корпоративних досліджень [4].

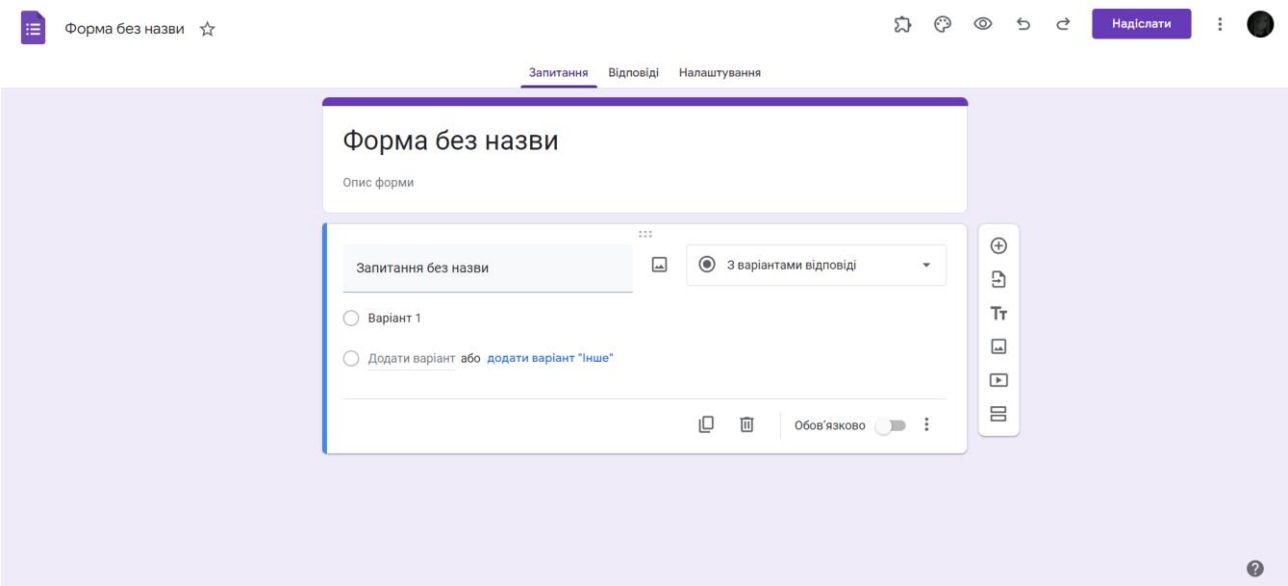

Рисунок 1.1 – Загальний вигляд інтерфейсу Google Forms

Переваги:

1. Безкоштовність: Google Forms є безкоштовним інструментом для всіх користувачів Google.

2. Інтеграція з іншими сервісами Google: Легко інтегрується з Google Sheets, що спрощує аналіз та обробку даних.

3. Простота використання: Інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати опитування навіть для користувачів без досвіду.

4. Можливість співпраці: Підтримує спільне редагування опитувань у реальному часі.

Недоліки:

1. Обмеженість функціоналу: Відсутність деяких розширених функцій, таких як логіка питань залежно від відповідей, обмеження в налаштуваннях дизайну та зовнішнього вигляду.

2. Обмежена аналітика: Менш потужні інструменти для аналізу даних порівняно з іншими професійними платформами.

SurveyMonkey (рис. 1.2) є ідеальним рішенням для бізнесів та організацій, які потребують професійних інструментів для проведення масштабних та комплексних опитувань [5].

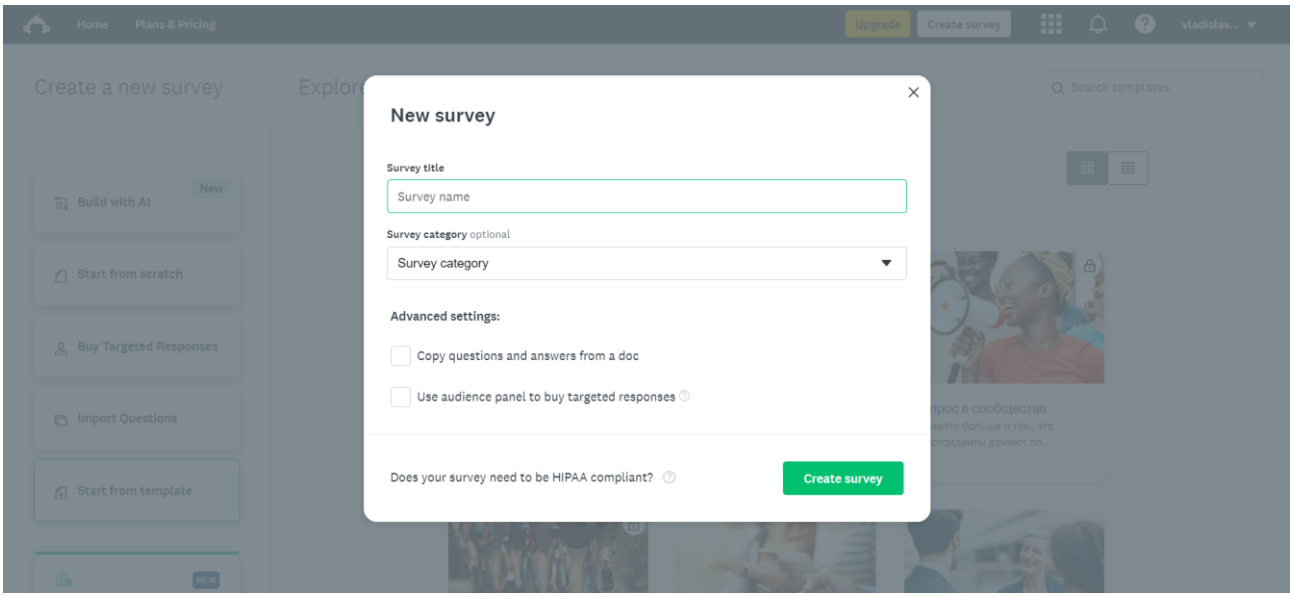

Рисунок 1.2 – Загальний вигляд інтерфейсу SurveyMonkey

Переваги:

1. Розширені можливості: Підтримує різні типи питань, логіку умовних переходів, можливість вставки зображень та відео.

2. Потужна аналітика: Вбудовані інструменти для аналізу даних, включаючи графіки, діаграми та можливість експорту результатів у різні формати.

3. Масштабованість: Підходить для великих опитувань з великою кількістю респондентів.

Недоліки:

1. Вартість: Платні тарифи можуть бути досить високими, особливо для малих бізнесів чи окремих користувачів.

2. Обмеження у безкоштовній версії: Безкоштовна версія має обмежений функціонал та максимальну кількість респондентів.

Typeform (рис. 1.3) підходить для маркетингових досліджень, опитувань клієнтів, а також для будь-яких випадків, де важливий привабливий зовнішній вигляд та користувацький досвід [6].

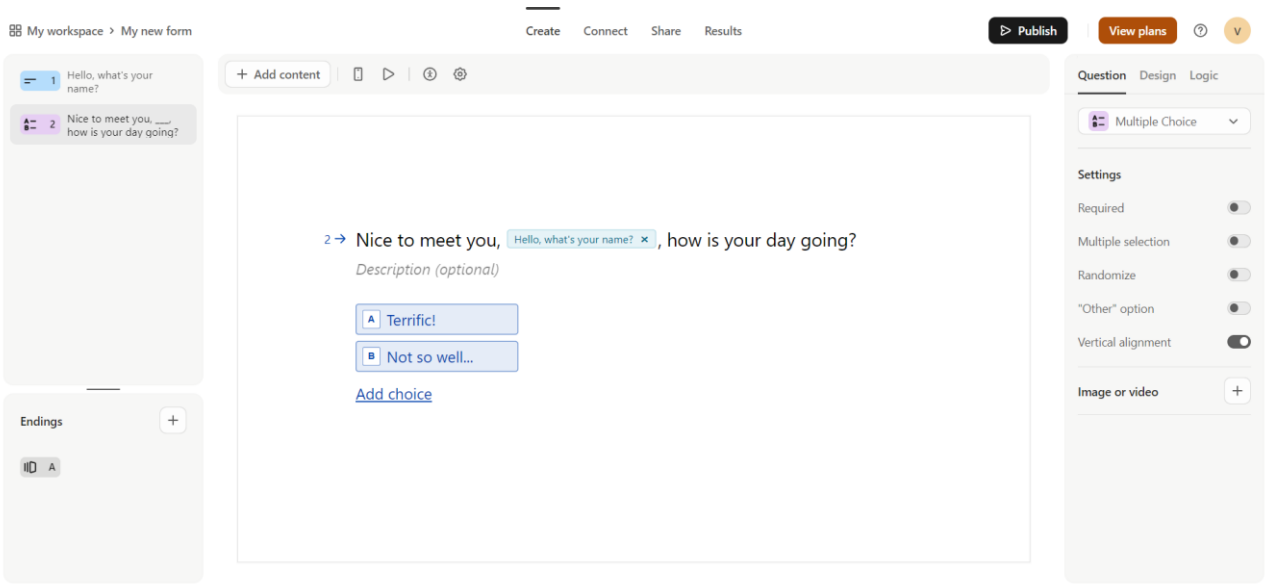

Рисунок 1.3 – Загальний вигляд інтерфейсу Typeform

Переваги:

1. Зручний та привабливий інтерфейс: Пропонує красиві та інтерактивні форми, що покращують залучення респондентів.

2. Гнучкість та налаштування: Можливість створення різноманітних типів питань, налаштування дизайну та брендування.

3. Інтеграція: Підтримка інтеграції з багатьма іншими інструментами, такими як Google Sheets, Mailchimp, Slack тощо.

Недоліки:

1. Вартість: Більшість корисних функцій доступні лише в платних планах.

2. Обмеження у безкоштовній версії: Обмежена кількість відповідей та базові функції у безкоштовному тарифі.

Qualtrics (рис. 1.4) є ідеальним рішенням для великих корпорацій, академічних установ та організацій, що потребують потужного та надійного інструменту для проведення комплексних опитувань та досліджень [7].

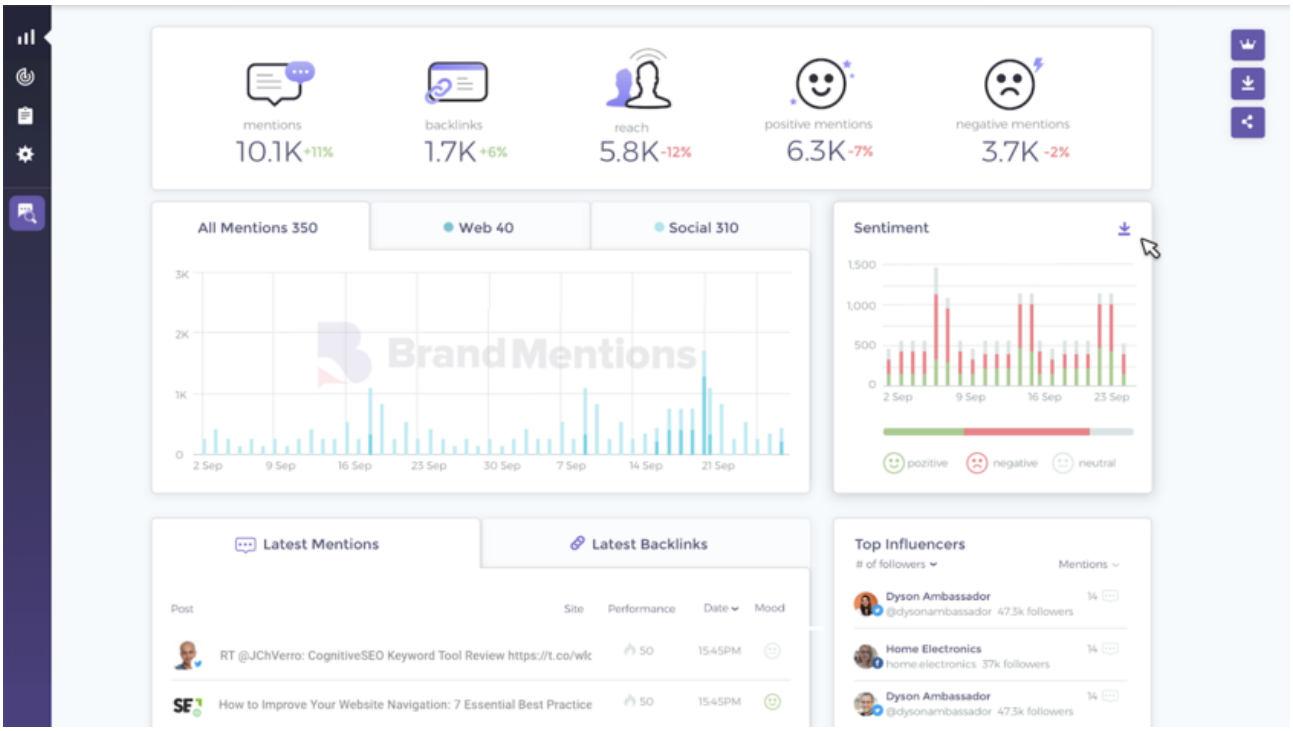

Рисунок 1.4 – Загальний вигляд інтерфейсу Qualtrics

Переваги:

1. Потужність та гнучкість: Широкий набір інструментів для створення та управління складними опитуваннями, включаючи логіку умовних переходів, різні типи питань та налаштування дизайну.

2. Аналітика та звітність: Потужні інструменти для аналізу даних, створення звітів та інтеграція з іншими системами.

3. Підтримка та безпека: Високий рівень технічної підтримки та безпеки даних.

Недоліки:

1. Вартість: Дуже висока вартість, що робить платформу недоступною для малих бізнесів та індивідуальних користувачів.

2. Складність: Може бути занадто складним для користувачів без технічного досвіду через велику кількість налаштувань та функцій.

У таблиці 1.1 наведено порівняльну характеристику обраних додатків для створення опитувань.

Таблиця 1.1 – Порівняльна характеристика популярних додатків для створення опитувань

Параметр / Додаток	Google Forms	SurveyMonkey	Typeform	Qualtrics
Доступність (плат./безкошт.)	Безкоштовний	Частково безкоштовний	Частково безкоштовний	Платний
Наявність статистики	Наявна	Наявна	Наявна	Наявна
Наявність інтеграції AI	Ні	Ні	Так	Так
Оцінка користувачів	4.5	4.2	4.6	4.4
Універсальність (комп./телефон)	Так	Так	Так	Ні
Наявність пробного періоду	Присутній	Присутній	Присутній	Присутній

Проаналізувавши обрані WEB-застосунки для проведення опитувань, можна дійти висновку, що кожен з них має свої особливості відносно типів, сктруктури та масштабності проведення опитувань, на основі чого можна визначити їх переваги та недоліки. Вибір конкретного застосунку залежить від потреб, бюджету, типу чи сфери проведення опитування та його масштабу.

1.3 Формулювання вимог та постановка задачі

Наразі, WEB-застосунки для проходження опитувань є, мабуть, найбільш популярним та зручним способом збору думок та відповідей респондентів стосовно певних питань і являють собою потужний інструмент для систематизації та аналізу експертних думок опитуваних користувачів. З їх

допомогою, будь-то з будь-якого куточку світу може висловити свою думку стосовно певних питань чи тематик у вигляді онлайн-опитування.

Для прикладу, Google Forms та Typeform є одними з найпопулярніших застосунків для створення та проходження опитувань, які надають можливість користувачам з усіх куточків світу доєднатися до онлайн-опитування чи створити власне з ціллю збору та аналізу статистичних даних стосовно тематики опитування, також гарантують анонімність респондентів. При чому, вони являються кросплатформними, тому ними можна скористатися на різних платформах і пристроях.

Проте, навіть такі масштабні та відомі WEB-застосунки мають свої недоліки та особливості використання, з переліку яких варто виокремити вартість послуг, не кожен застосунок є повністю безкоштовним, або пропонує обмежений функціонал, при використанні безкоштовної версії; обмеженість варіантів відповідей на питання опитувань, наприклад, відсутність можливості прикріпити медіа-, аудіо-файл як варіант відповіді; обмеження кількості питань чи респондентів; обмеження відображення повних статистичних даних тощо.

Загалом, WEB-застосунки для проходження опитувань надають користувачам широкий діапазон можливостей в створенні та проходженні опитувань, збору, аналізу та систематизації відповідей респондентів, проте , при виборі потрібного застосунку важливо враховувати не тільки його переваги, а й недоліки.

WEB-застосунок для проходження опитувань міститиме наступний спектр функцій:

- Можливість створювати власний обліковий запис, вказавши логін, email та пароль;
- Можливість створювати та редагувати створені опитування;
- Можливість проходити власні створені опитування та опитуваннях інших користувачів;
- Можливість переглядати статистику пройдених опитувань.

Під час виконання роботи, необхідно створити WEB-застосунок для проходження опитувань. WEB-застосунок дозволятиме користувачам створювати, редагувати та проходити опитування.

Користувацький інтерфейс WEB-застосунку повинен бути інтуїтивно зрозумілим для користувачів, поля вводу, навігаційні елементи та інші частини інтерфейсу мають бути помітними та зручними у використанні, при чому вихідні дані повинні подаватись користувачеві у зрозумілому вигляді.

Розроблюваний WEB-застосунок повинен бути зручним у використанні, тобто, користувач повинен мати можливість переглянути створені та пройдені опитування, редагувати власні опитування, переглядати статистику пройдених та створених опитувань.

Важливою частиною подібних WEB-застосунків є наявність локалізації декількома мовами. Користувач повинен мати можливість змінити мову перекладу сайту на зручну для нього, що робить застосунок зручним у використанні і для іноземців.

1.4 Висновок

В даному розділі було проаналізовано існуючі платформи для розгортання WEB-застосунку для проходження опитувань та обрано найбільш підходящу – WEB-ресурс, беручи до уваги його зручність, функціональність, кросплатформність та універсальність. Проведено аналіз існуючих застосунків-аналогів, при чому виявлено певні недоліки в них, головними з яких можна було б виділити платність, або не повний функціонал при використанні пробної версії, що доволі сильно обмежує можливість користувачів, які не готові оформлювати платну підписку; обмеженість у виборі варіантів відповідей на опитування, їх формат. При цьому, навіть у таких масштабних ресурсів іноді можуть виникати проблеми з конфіденційністю, оскільки вони зачасту можуть використовувати особисті дані користувачів в маркетологічних чи рекламних цілях. Проте, не зважаючи підтримку різних пристроїв, на деяких з ним, вони можуть надавати врізаний функціонал.

WEB-застосунок для проведення опитувань є доволі актуальним, оскільки він сприятиме значному спрощенню процесу проведення різного виду опитувань у багатьох галузях для збору і систематизації думок респондентів з приводу різноманітних питань в режимі реального часу. При чому, він здатний допомогти людям в різноманітних цілях, наприклад, його можна використати в навчанні для оцінки розробленого студентом програмного забезпечення, який показав би його доцільність на основі думок таких же студентів чи викладачів. Саме тому створення даного WEB-застосунку є доцільним.

Виконано постановку задачі та опис призначення розроблюваного WEB-застосунку, внаслідок чого – було поставлено задачі для подальшого дослідження.

2 ПРОЕКТУВАННЯ WEB-ЗАСТОСУНКУ ДЛЯ ПРОХОДЖЕННЯ ОПИТУВАНЬ

2.1 Розробка структури WEB-застосунку для проходження опитувань

Подібно до десктопних додатків, WEB-застосунок складається з окремого набору визначених компонентів, розбитих на окремі файли. Структура WEB-застосунку базується на розподілі компонентів на окремі функціональні модулі, кожен з яких виконує свої унікальні функції стосовно програмної реалізації застосунку. Яскравим прикладом модулів застосунку можуть бути: головна сторінка застосунку; модуль реалізації зміни локалізації; сторінка проходження, створення, редагування, результатів опитування; адміністративні чи функціональні панелі тощо.

Кожен модуль WEB-застосунку розробляється незалежно, що гарантує автономність та незалежність від інших модулів застосунку. Такий підхід дозволяє паралельно розробляти WEB-застосунок по частинах, зосереджуючись на конкретній задачі у конкретний час, без необхідності якось спеціалізовано змінювати вже існуючий програмний код застосунку, що в свою чергу дає змогу більш зручно та ефективно масштабувати WEB-застосунок в майбутній розробці.

Кожен створений модуль WEB-застосунку має свою унікальну назву та структуру файлів, що спрощує розуміння та організацію програмного коду застосунку. Окремий модуль може містить власні похідні файли для CSS-стилів оформлення графічного інтерфейсу, HTML-шаблони загальної структури сторінки, JavaScript-скрипти для керування логікою та рендерингу окремих компонентів, спеціалізованих зображень тощо. Використання модульної структури WEB-застосунку полегшує його розвиток і підтримку у майбутньому, дозволяє легше вносити необхідні правки в окремі модулі, робить саму структуру більш зрозумілою.

WEB-застосунок для проходження опитувань міститиме наступні сторінки: сторінка для логіну/реєстрації, особистий кабінет, сторінка для

доєднання опитування, сторінка перегляду створених та пройдених опитувань, сторінки створення, редагування, проходження та результатів опитування. Кожна сторінка міститиме модулі графічного оформлення, розмітки, функціоналу клієнтської та серверної частини [8], остання безпосередньо взаємодіятиме з базою даних. Структура WEB-застосунку для проходження опитувань зображена на рис. 2.1.

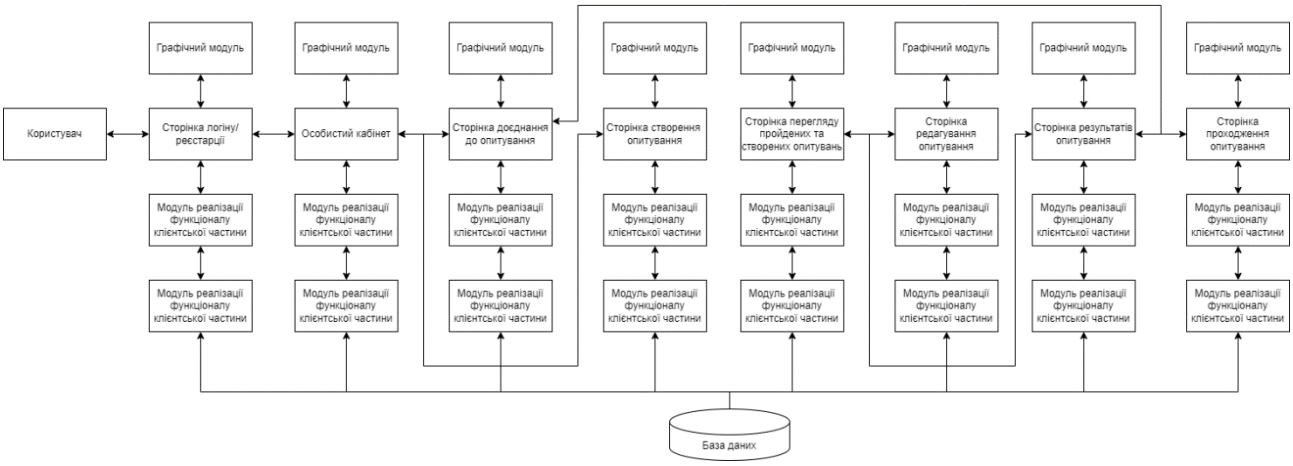

Рисунок 2.1 − Загальна структура WEB-застосунку для проходження опитувань

На сторінці логіну/реєстрації користувач може зайти в свій обліковий запис, або зареєструвати новий. Для логіну користувач має ввести свій пароль та електронну адресу та натиснути кнопку підтвердження форми або використати спеціальну Google авторизацію; для реєстрації нового облікового запису, користувач має ввести ім’я користувача, електронну адресу та придумати пароль, після чого переключити сторінку на логін та увійти, або використати спеціальну Google авторизацію. Після проведених маніпуляцій, дані користувача передаватимуться на сервер для обробки [9].

Увійшовши в особистий кабінет користувач може рухатись у трьох напрямках: доєднатися до опитування, переглянути створені та пройдені опитування, створити нове опитування, за що відповідають окремі навігаційні кнопки, а також користувач матиме можливість вийти з облікового запису за допомогою кнопки LogOut.

Відкривши сторінку створення опитування користувач матиме змогу створити власне опитування. Для створення опитування, необхідно придумати назву та опис опитування і ввести їх у відповідні поля вводу, наступний крок – створення питань опитування. Щоб додати нове питання потрібно натиснути кнопку додавання нового питання, після чого потрібно заповнити його за шаблоном: обрати тип питання, яким може бути текст, фото та відео, при чому фото та відео подаються у форматі посилань на ресурси, ввести необхідне питання чи посилання, обрати спектр емоції з пропонованих стандартний чи додати ще й власні. Провівши таку маніпуляцію з n-ю кількістю питань, для створення опитування необхідно натиснути кнопку підтвердження створення опитування, після успішного створення відкриється сторінка з підтвердженням створення опитування, на якій міститиметься назва та опис опитування, унікальні ID та код опитування, а також посилання на нього.

На сторінці доєднання до опитування користувач матиме можливість доєднатися до опитування, використовуючи унікальний код опитування, який можна отримати про його створенні.

На сторінці проходження опитування користувач має відповісти на усі питання опитування. Кожне питання має свій тип: текст, фото і відео; спектр емоцій для оцінки, відповідь на які подається у форматі повзунків з градацією від 0 до 100 на кожну емоцію для кожного питання; назва та опис опитування подано перед усіма питаннями. Після заповнення спектрів емоцій для кожного питання у вигляді повзунків, користувач має натиснути кнопку підтвердження надсилання форми, після чого застосунок надішле запит з даними форми до сервера для збереження відповіді у базі даних; користувача переправить у особистий кабінет. Після цього, щоб переглянути статистику пройденого опитування, користувач може перейти на сторінку перегляду створених та пройдених опитувань, переключити сторінку у формат перегляду пройдених опитувань, знайшовши опитування за назвою потрібно буде натиснути кнопку результатів.

Перейшовши на сторінку перегляду створених та пройдених опитувань, користувач матиме змогу переглянути списки створених та пройдених

опитувань. Для зміни опитувань на створені та пройдені користувач має натиснути на відповідну кнопку в верхній частині тіла сторінки, після чого вони будуть відображені в основній частині сторінки. При перегляді створених користувачем опитувань відображатимуться створені опитування, де біля назви кожного опитування користувач матиме змогу скопіювати посилання на опитування, редагувати його чи переглянути статистику проходження опитування. При перегляді пройдених користувачем опитувань відображатимуться пройдені опитування, при чому користувач матиме зможу перевірити їх статистику проходження.

На сторінці редагування опитування користувач матиме змогу змінити, створені ним раніше, опитування. Під час редагування опитування користувач може змінити назву та опис опитування, редагувати, додавати та видаляти питання згідно алгоритму як при його створенні. Для успішного редагування опитування користувачеві необхідно зберегти всі питання з допомогою відповідних кнопок та повністю зберегти все опитування за допомогою кнопки збереження опитування.

На сторінці перегляду результатів опитування користувач може ознайомитись з загальною статистикою проходження опитування, яка базується на відповідях усіх користувачів у вигляді спеціалізованих графіків загальних оцінок спектрів емоцій, медіани оцінки тощо.

На рисунку 2.2 зображено структуру принципу роботи WEB-застосунку для проходження опитувань, з чого можна побачити послідовність виконання запиту користувача і отримання необхідної відповіді від бази даних. При виконанні певних дій в застосунку, формується запит користувача, який спочатку обробляється за допомогою JS і в обробленому форматі відправляється для обробки на сервер, який в свою чергу приймає запит у поданому форматі і формує запит до бази даних для отримання потрібної інформації. Після отримання відповіді від бази даних відбувається зворотній процес, який полягає у надсиланні результатів запиту користувачеві у форматі відповіді WEB-застосунку.

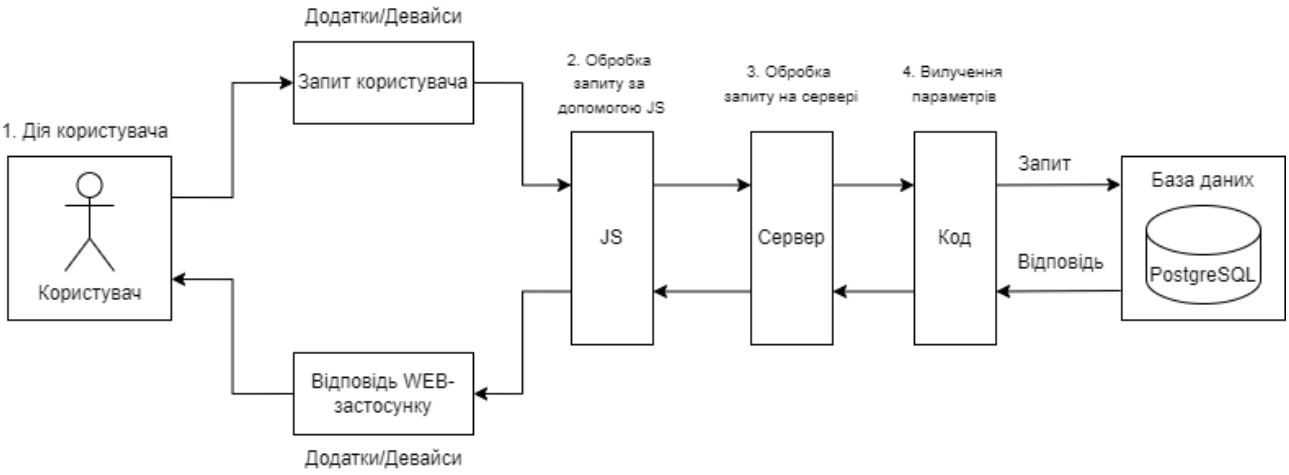

Рисунок 2.2 – Структура принципу роботи WEB-застосунку для проведення опитувань

2.2 Розробка UML-діаграми класів

У мові програмування JavaScript розробка програм має змогу базуватись на основі використання об’єктно-орієнтованого підходу (ООП).

Об'єктно-орієнтоване програмування (ООП) базується на поняттях «клас» і «об'єкт». При використанні ООП програміст структурує задачу на логічно завершені одиниці, які реалізуються через класи. Кожен клас є шаблоном або описом, за допомогою якого створюються об'єкти - конкретні екземпляри класу [10].

Підхід ООП передбачає, що програміст повинен вміти:

- Розбити задачу на класи, які виконують конкретні функції або мають певну відповідальність;
- Встановлювати та побудувати зв'язки між класами у програмі для досягнення потрібного функціоналу та ефективної комунікації;
- Виділяти та реалізовувати багаторазово використовуваний код у вигляді окремих класів для зменшення дублювання коду та дотримання принципу DRY [10];
- Створювати ієрархію між класами за принципом «від простого до складного» для впорядкування класів за рівнем абстракції та повторного використання коду [10];

- Розуміти деталі реалізації можливостей, які надають класи для ефективної побудови програм: інкапсуляція, спадковість, поліморфізм [10].

Розроблюваний програмний застосунок міститиме наступні класи: LoginPage, LoggedInPage, SurveyCreationPage, TakeSurveyPage, SurveyPage, MySurvey, EditSurvey, SurveyResults що слугуватимуть для відображення змісту WEB-сторінок.

Клас LoginPage містить наступні методи: toggleMode(), ForgotPassword(), handleLogin(e), handleSignUp(e); атрибути класу: isLogin, name, email, password, message. Атрибут isLogin це стан для перемикання між входом і створенням облікового запису; атрибут name це стан для поля введення імені; атрибут email це стан для поля введення електронної пошти; атрибут password це стан для поля введення паролю; атрибут message це стан для відображення повідомлень користувачеві. Метод toggleMode() перемикає режим між входом і реєстрацією, очищує поля введення; метод ForgotPassword() використовується для відновлення паролю; метод handleLogin(e) обробляє подію входу користувача, перевіряє введені дані, надсилає запит на сервер; метод handleSignUp(e) обробляє подію реєстрації користувача, перевіряє введені дані, надсилає запит на сервер.

Клас LoggedInPage містить наступні методи: createSurvey(), takeSurvey(), mySurveys(), logout(); атрибути класу: userInfo. Атрибут userInfo це стан для збереження інформації про користувача. Метод createSurvey() створює нове опитування і перенаправляє на сторінку створення опитування; метод takeSurvey() дозволяє користувачу пройти опитування і перенаправляє на відповідну сторінку; метод mySurveys() показує користувачу його опитування і перенаправляє на сторінку з опитуваннями; метод logout() видаляє токен аутентифікації з локального сховища і перенаправляє на головну сторінку.

Клас SurveyCreationPage містить наступні методи: validateForm(), handleSubmit(), handleAddQuestion(); атрибути класу: questions, quizTitle, quizDescription. Атрибут questions це стан для збереження списку питань; атрибут quizTitle це стан для збереження заголовка опитування; атрибут quizDescription це стан для збереження опису опитування. Метод validateForm() перевіряє, чи всі текстові питання заповнені, чи всі URL-адреси для зображень і

відео валідні, і чи вибрано хоча б одну емоцію для кожного питання; метод handleSubmit() перевіряє валідність форми, генерує хеш опитування, отримує токен аутентифікації з локального сховища і надсилає POST запит на сервер для створення нового опитування; метод handleAddQuestion() додає нове питання до списку питань.

Клас TakeSurveyPage містить наступні методи: handleSubmit(); атрибути класу: surveyHash. Атрибут surveyHash це стан для збереження ідентифікатора опитування. Метод handleSubmit(event) обробляє подію надсилання форми, перевіряє ідентифікатор опитування та перенаправляє користувача на сторінку з опитуванням.

Клас SurveyPage містить наступні методи: handleEmotionChange (questionId, emotion, value), submitSurvey(); атрибути класу: surveyHash, survey, emotions, isLoading, error, submitte. Атрибут surveyHash це змінна, що зберігає ідентифікатор опитування; атрибут survey це стан для збереження даних опитування; атрибут emotions це стан для збереження емоційних відповідей на питання опитування; атрибут isLoading це стан для відображення стану завантаження; атрибут error це стан для збереження повідомлення про помилку; атрибут submitted це стан для збереження інформації про те, чи було опитування надіслано. Метод handleEmotionChange(questionId, emotion, value) оновлює стан спектру емоцій для певного питання; метод submitSurvey() надсилає відповідь на сервер і перенаправляє користувача на сторінку особистого кабінету.

Клас MySurvey містить наступні методи: fetchData, copyToClipboard(text), navigateTo(url); атрибути класу: activeTab, createdSurveys, takenSurveys. Атрибут activeTab це стан для збереження активної вкладки ('created' або 'taken'); атрибут createdSurveys це стан для збереження опитувань, створених користувачем; атрибут takenSurveys це стан для збереження опитувань, в яких користувач брав участь. Метод fetchData() виконує запит до сервера для отримання опитувань на основі активної вкладки; метод copyToClipboard(text) копіює переданий текст до буферу обміну з використанням нового API буфера обміну або запасного методу з використанням тимчасової текстової області; метод navigateTo(url) перенаправляє користувача на потрібну сторінку.

Клас EditSurvey містить наступні методи: fetchSurvey(), handleSave(), handleQuestionChange(index, key, value), handleSaveQuestion(questionId, question); атрибути класу: id, survey, title, description, questions, message, questionMessage. Атрибут id зберігає ідентифікатор опитування; атрибут survey використовується для збереження даних опитування; атрибут title використовується для збереження назви опитування; атрибут description використовується для збереження опису опитування; атрибут questions використовується для збереження питань опитування; атрибут message використовується для збереження повідомлення про збереження змін; атрибут questionMessage використовується для збереження повідомлення про збереження змін питань. Метод fetchSurvey() виконує запит до API для отримання даних опитування та оновлює відповідні стани; метод handleSave() надсилає оновлені дані опитування на сервер і встановлює повідомлення про успішне збереження; метод handleQuestionChange(index, key, value) оновлює стан питань опитування; метод handleSaveQuestion(questionId, question) надсилає оновлені дані конкретного питання на сервер і встановлює повідомлення про успішне збереження.

Клас SurveyResults містить наступні методи: fetchResults(), fetchAllResults(), prepareChartData(), prepareAverageData(), renderContent(type, content); атрибути класу: id, data, allData. Атрибут id зберігає ідентифікатор опитування; атрибут data зберігає дані результатів опитування; атрибут allData зберігає дані всіх результатів опитування для порівняння. Метод fetchResults() виконує запит до сервера для отримання даних результатів конкретного опитування; метод fetchAllResults() виконує запит до сервера для отримання всіх результатів опитування для порівняння; методи prepareChartData() та prepareAverageData() використовуються для підготовки даних для графіків; метод renderContent(type, content) відображає вигляд змісту питання в залежності від його типу.

На рис. 2.3 зображено UML-діаграму класів WEB-застосунку для проходження опитувань.

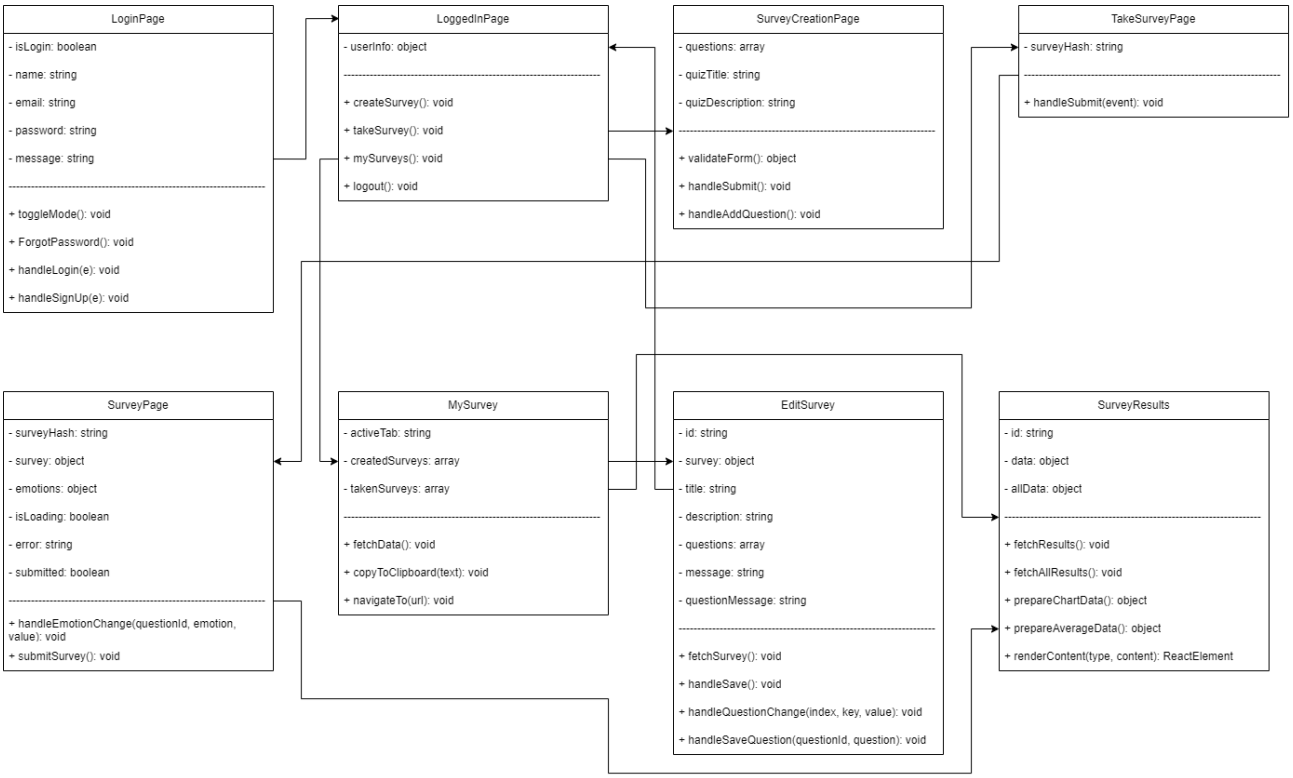

Рисунок 2.3 – UML-діаграма класів WEB-застосунку для проходження опитувань

2.3 Розробка схеми алгоритму роботи WEB-застосунку для проходження опитувань

Алгоритм − це скінчена послідовність указівок на виконання дій, спрямованих на розв’язування задачі. Алгоритм може бути візуалізований за допомогою блок-схем. Для комп'ютерних програм алгоритм є списком деталізованих інструкцій, що реалізують процес обчислення.

Опис алгоритму може бути представлений у формі псевдокоду, блок-схеми, програмного коду або простого тексту, залежно від контексту та специфіки задачі.

Псевдокод: Це спеціальний мовний вигляд, який нагадує програмний код, але не повністю синтаксично правильний. Псевдокод зазвичай використовується для передачі загальної логіки алгоритму та послідовності операцій.

Блок-схема: Це графічне зображення алгоритму за допомогою блоків та зв'язків між ними. Блок-схема дозволяє візуалізувати послідовність дій, умови та повторюваність.

Програмний код: Алгоритм може бути представлений у вигляді реального програмного коду для певної мови програмування. Код залежить від мови програмування, але має точність та синтаксис для виконання певних дій.

Отже, для розробки WEB-застосунку для проходження опитувань було обрано блок-схему опису алгоритму. Схема алгоритму роботи WEB-застосунку для проходження опитувань зображена на рис. 2.4.

І випадок

Алгоритм у випадку натиснення на кнопку «Реєстрації» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи логін;

Крок 2. Користувач натискає кнопку «Реєстрація»;

Крок 3. Введення вхідних даних: Ім’я користувача, email, пароль;

Крок 4. Створення нового запису у базі даних.

ІІ випадок

Алгоритм у випадку натиснення на кнопку «Логін» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи Логін;

Крок 2. Користувач натискає кнопку «Логін»;

Крок 3. Введення вхідних даних: Email, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 5. Вхід у систему.

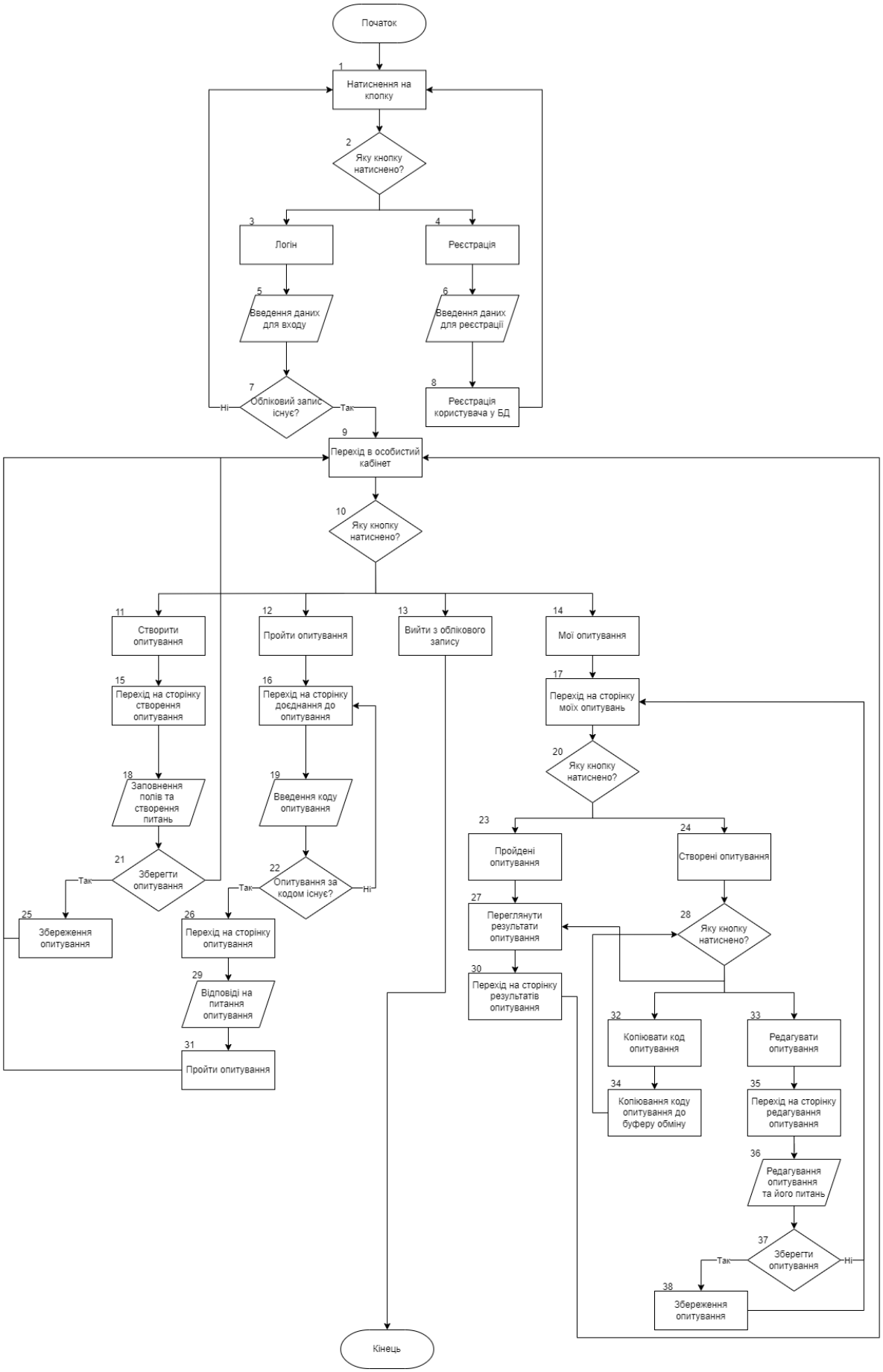

Рисунок 2.4 – Схема алгоритму роботи WEB-застосунку для проходження опитувань

III випадок

Алгоритм у випадку «Створення опитування» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи логін;

Крок 2. Користувач натискає кнопку «Логін»;

Крок 3. Введення вхідних даних: Email, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 5. Вхід у систему;

Крок 6. Користувач обирає «Створення опитування»;

Крок 7. Введення назви та опису опитування;

Крок 8. Формування та заповнення пинать згідно шаблону;

Крок 9. Користувач натискає «Створити опитування»;

Крок 10. Надсилання запиту до сервера для збереження опитування у БД.

IV випадок

Алгоритм у випадку «Проходження опитування» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи логін;

Крок 2. Користувач натискає кнопку «Логін»;

Крок 3. Введення вхідних даних: Email, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 5. Вхід у систему;

Крок 6. Користувач обирає «Долучитися до опитування»;

Крок 7. Перехід на сторінку доєднання до опитування;

Крок 8. Користувач вводить унікальний код опитування;

Крок 9. Користувач натискає «Пройти опитування»;

Крок 10. Перехід до сторінки проведення опитування;

Крок 11. Проходження опитування;

Крок 12. Користувач натискає «Підтвердити проходження опитування»;

Крок 13. Надсилання запиту до сервера для збереження результатів опитування у БД.

V випадок

Алгоритм у випадку «Редагування опитування» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи логін;

Крок 2. Користувач натискає кнопку «Логін»;

Крок 3. Введення вхідних даних: Email, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 5. Вхід у систему;

Крок 6. Користувач обирає «Мої опитування»;

Крок 7. Перехід на сторінку опитувань користувача;

Крок 8. Користувач обирає «Створені опитування»;

Крок 9. Користувач обирає потрібне опитування і натискає «Редагувати опитування»;

Крок 10. Перехід на сторінку редагування опитування;

Крок 11. Користувач редагує опитування;

Крок 12. Користувач натискає «Зберегти опитування»;

Крок 13. Надсилання запиту до сервера для збереження опитування у БД.

VI випадок

Алгоритм у випадку «Результати опитування» складається з таких кроків:

Крок 1. На першому етапі користувач входить до системи та обирає реєстрацію чи логін;

Крок 2. Користувач натискає кнопку «Логін»;

Крок 3. Введення вхідних даних: Email, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 5. Вхід у систему;

Крок 6. Користувач обирає «Мої опитування»;

Крок 7. Перехід на сторінку опитувань користувача;

Крок 8. Користувач обирає «Пройдені опитування» або «Створені опитування»;

Крок 9. Користувач обирає потрібне опитування і натискає «Результати опитування»;

Крок 10. Перехід на сторінку результатів опитування.

2.4 Висновок

У даному розділі проаналізовано та сформовано основні вимоги до програмного забезпечення.

Розроблено структуру WEB-застосунку для проходження опитувань, за допомогою якої можна значно спростити розробку застосунку. WEB-застосунок міститиме наступні сторінки: сторінка для логіну/реєстрації, особистий кабінет, сторінка для доєднання опитування, сторінка перегляду створених та пройдених опитувань, сторінки створення, редагування, проходження та результатів опитування.

Створено UML-діаграму класів та описано її класи, методи та атрибути. Використання UML-діаграм класів дозволяє візуалізувати структуру програми та методи, які присутні в кожному класі. Це надає швидкий та зручний спосіб отримати інформацію про класи та їх взаємозв'язки. За допомогою цих діаграм можна аналізувати структуру системи, розкривати деталі та проектувати програму з точки зору об'єктно-орієнтованого підходу. Розробка діаграм класів є важливою складовою процесу розробки програмного забезпечення. Вона дозволяє провести декомпозицію предметної галузі, визначити необхідні класи та їх взаємозв'язки. Це допомагає краще розуміти сутність проблеми та визначати, які аспекти треба враховувати, а які можна ігнорувати.

Крім того, розроблено структуру принципу роботи та схему алгоритму функціонування WEB-застосунку для проходження опитувань. Схема алгоритму допомагає крок за кроком уявити, як програма буде працювати та взаємодіяти з іншими елементами, що сприяє кращому розумінню задачі та уникненню помилок під час реалізації функціоналу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ ДЛЯ ПРОХОДЖЕННЯ ОПИТУВАНЬ

3.1 Обґрунтування вибору мови програмування та бібліотек

Найпопулярнішими мовами програмування для розробки WEB-додатків є Python, JavaScript та C#. Кожна з цих мов використовується в певному середовищі та має свої переваги та недоліки.

Python – мова програмування, що підтримує одразу декілька парадигм програмування, такі як процедурна, об’єктно-орієнтована та функціональна парадигми. Вона є основною мовою для фреймворків Flask, Django та ін., які широко використовується для створення веб-додатків [11].

JavaScript – мова програмування, що є ключовою для розробки WEB-застосунків. Вона підтримує об’єктно-орієнтовану та функціональну парадигми програмування і використовується в фреймворках React для розробки користувацького інтерфейсу та Node.js для серверної частини [12].

C# – мова програмування, що підтримує об’єктно-орієнтовану парадигму програмування. Вона використовується в середовищі .NET, яке забезпечує широкий спектр інструментів для розробки різноманітних додатків, від WEB-застосунків до корпоративних систем [13].

У таблиці 3.1 наведено основні відмінності між вищезгаданими мовами програмування.

Таблиця 3.1 – Порівняння мов програмування

Назва критерію	Python	JavaScript	C#
Динамічна типізація	+	+	+
Неявне приведення до типів	+	+	+
Псевдоніми типів	+	+	-
Виведення сигнатури для локальних методів	-	+	-
Багатопоточна компіляція	-	+	+

Таблиця 3.1 – Продовження

Назва критерію	Python	JavaScript	C#
Створення об’єктів у стеку	+	+	+
Спискові виключень	+	+	+
Значення параметрів за замовчуванням	+	+	+

Динамічна типізація дозволяє створювати динамічні методи та класи, тип яких буде визначений під час виконання роботи програми, а також використовувати один і той самий метод для різних типів даних.

Неявне приведення до типів дозволяє при наслідуванні конвертувати дані з типів наслідуваних класів до конкретних реалізацій та навпаки. Це особливо зручно при роботі з колекціями різнотипних даних.

Псевдоніми типів дозволяють перевіряти тип даних об’єкта під час виконання програми. Ця функція є дуже зручною разом з динамічною типізацією, оскільки за рахунок перевірок типу можна значно зменшити кількість методів-дублікатів.

Виведення сигнатури для локальних методів дозволяє динамічно, під час виконання програми, видаляти та змінювати сигнатуру локальних методів. Ця функція притаманна функціональним мовам програмування.

Багатопоточна компіляція підвищує продуктивність шляхом одночасної обробки кількох потоків даних.

Створення об’єктів у стеку дозволяє ефективно керувати пам’яттю, зберігаючи дані у стеку, що сприяє швидкому доступу до них.

Спискові виключення допомагають компактно описати чергу подій, необхідних для виконання. У Python та JavaScript ці списки можуть бути реалізовані через обробку виключень, делегати та події.

Значення параметрів за замовчуванням дозволяють викликати один і той самий метод з різною кількістю параметрів, без необхідності кожен раз передавати ці ж самі параметри.

Оцінивши основні відмінності між мовами програмування для WEB-розробки, для розробки WEB-застосунку для проходження опитувань, було обрано мову програмування JavaScript.

Серед найпопулярніших фреймворків для створення веб-застосунків можна виокремити:

- Django (Python);
- Flask (Python);
- Node.js з Express (JavaScript);
- React (JavaScript);
- ASP.NET Core (C#);
- Blazor (C#).

Django – потужний фреймворк для веб-розробки з акцентом на швидкий розвиток та чистий, прагматичний дизайн, який має вбудовану систему аутентифікації, ORM для роботи з базами даних, можливість створення адмін-панелі, велика спільнота та багато бібліотек. Ідеально підходить для створення складних веб-додатків з великою кількістю функцій [14].

Flask – легкий мікрофреймворк, який надає лише основні компоненти для веб-розробки, простий у використанні, гнучкий, можлива інтеграція з різними бібліотеками. Підходить для невеликих та середніх проектів, де важлива гнучкість та мінімалізм [15].

Node.js з Express: Node.js – це середовище виконання JavaScript, Express – це мінімалістичний веб-фреймворк для Node.js. Основні його переваги - висока продуктивність, легка масштабованість, велика кількість доступних модулів. Ідеально підходить для створення серверних частин веб-додатків, включаючи API для опитувань [16].

React – це фреймворк для створення користувацьких інтерфейсів, розроблений компанією Facebook. Основні його переваги - висока продуктивність, зручність компонування, велика спільнота та багато бібліотек. Підходить для створення фронтенд частини веб-додатків, зокрема інтерфейсів для опитувань [17].

ASP.NET Core – кросплатформний фреймворк для веб-розробки від Microsoft. Основні його переваги - висока продуктивність, підтримка масштабованих рішень, інтеграція з іншими продуктами Microsoft, сильна типізація. Підходить для створення великих корпоративних веб-додатків, включаючи системи для проведення опитувань.

Blazor - фреймворк для створення інтерактивних веб-додатків з використанням мови програмування C# без використання JavaScript. Основні його переваги - можливість використання C# для клієнтської частини, повна інтеграція з екосистемою .NET, продуктивність. Ідеально підходить для розробників, які хочуть використовувати C# як для серверної, так і для клієнтської частини.

У таблиці 3.2 наведено порівняння вищезгаданих фреймворків.

Таблиця 3.2 – порівняння фреймворків для розробки WEB-застосунків

Фреймворк	Мова програмування	Легкість навчання	Продуктивність	Гнучкість	Інтеграція з сервісами
Django	Python	Важка	Висока	Середня	Середня
Flask	Python	Легка	Середня	Висока	Висока
Node.js Express	JavaScript	Легка	Висока	Висока	Висока
React	JavaScript	Легка	Висока	Висока	Висока
ASP.NET Core	C#	Середня	Висока	Середня	Висока
Blazor	C#	Середня	Висока	Середня	Висока

Легкість навчання визначає, наскільки просто новачкам або розробникам з досвідом в інших мовах програмування освоїти нову технологію або фреймворк.

Продуктивність фреймворку відображає, наскільки ефективно він використовує ресурси системи (процесор, пам'ять) і наскільки швидко виконує завдання.

Гнучкість фреймворку визначає, наскільки він дозволяє адаптуватися до різних вимог і умов, наскільки легко його можна розширити або налаштувати під специфічні потреби.

Інтеграція з іншими сервісами визначає, наскільки просто фреймворк дозволяє підключати сторонні сервіси, бібліотеки та API для розширення функціональності додатку.

Порівнюючи фреймворки для створення веб-застосунку для проведення опитувань, можна зробити висновок, що React є оптимальним вибором для розробки клієнтської частини застосунку завдяки його високій продуктивності, зручності компонування та великій спільноті. React дозволяє створювати інтерфейси, які легко масштабуються та інтегруються з іншими сервісами, що є важливим для створення інтерактивного та ефективного WEB-застосунку для проходження опитувань.

3.2 Обгрунтування вибору середовища розробки

IDE які можна використати для розробки:

- Visual Studio;
- Visual Studio Code;
- JetBrains Rider.

Visual Studio − це інтегроване середовище розробки для розробки додатків із використанням мови програмування .NET та інших мов програмування, таких як C++, Python, JavaScript тощо. Платформа Visual Studio складається з компонентів і призначена для розширення за допомогою додаткових компонентів. Розроблена з використанням .NET, платформа Visual Studio може використовуватися для розробки WEB-, хмарних, настільних, мобільних і ігрових додатків, інтегрованих середовищ розробки та інших інструментів. Visual Studio можна використовувати як IDE для будь-якої мови програмування, для якої доступний компонент [18].

Visual Studio Code − це редактор вихідного коду, який працює на Windows, macOS і Linux. Visual Studio Code підтримує різні мови програмування, такі як

JavaScript, TypeScript, Python, C#, Java і інші. Редактор Visual Studio Code складається з розширень і призначений для розширення за допомогою додаткових розширень. Розроблений з використанням Electron, редактор Visual Studio Code може використовуватися для редагування, налагодження і запуску коду для різних платформ і пристроїв. Visual Studio Code можна використовувати як редактор для будь-якої мови програмування, для якої доступне розширення [19].

JetBrains Rider − це кросплатформне інтегроване середовище розробки (IDE) для розробки додатків із використанням мови програмування .NET та інших мов програмування, таких як C#, F#, VB.NET тощо. Rider базується на платформі IntelliJ і включає функціональність ReSharper, яка надає потужні можливості для аналізу, редагування, рефакторингу і налагодження коду. Rider підтримує різні типи проектів .NET, такі як .NET Framework, .NET Core, Mono, Xamarin, Unity, ASP.NET і ASP.NET Core. Rider працює на Windows, macOS і Linux [20].

Отже, після аналізу IDE, враховуючи усі переваги та недоліки було обрано середовище для розробки − Visual Studio Code, оскільки вона зручна, швидка, має безліч функцій, які допомагають прискорити розробку.

3.3 Розробка клієнтської частини

Для розробки програмного модуля використано наступні технології:

React — це відкритий JavaScript-фреймворк, а точніше, бібліотека JavaScript, яка використовується для розробки інтерфейсів користувача. React дозволяє ефективно створювати застосунки з високою продуктивністю і масштабованістю. Одним з ключових концепцій у React JS є компоненти. Вони представляють собою незалежні блоки коду, які відповідають за рендеринг певної частини користувацького інтерфейсу. React використовується для розробки клієнтської частини програмного модуля [11].

HTML − це мова розмітки для створення WEB-сторінок. HTML складається з низки елементів, які вказують браузеру, як відображати контент.

HTML також дозволяє створювати гіперпосилання, які з'єднують WEB -сторінки одну з одною, або з іншими ресурсами в Інтернеті. HTML використовується для розробки клієнтської частини програмного модуля [21].

JavaScript − це мова програмування для створення динамічного та інтерактивного контенту на WEB-сторінках, такого як анімація, графіка, форми, валідація даних тощо. JavaScript також може спілкуватися з сервером за допомогою AJAX або WebSockets для отримання або надсилання даних без перезавантаження сторінки. JavaScript може працювати разом з іншими мовами та технологіями, такими як HTML, CSS, XML, JSON та багатьма іншими. JavaScript використовується для розробки клієнтської частини програмного модуля [21].

CSS − це мова для стилізації HTML-документів. CSS описує, як HTML-елементи мають відображатися на екрані, папері або в інших медіа. CSS використовується для розробки клієнтської частини програмного модуля [21].

Розроблюваний програмний застосунок міститиме наступні класи: LoginPage, LoggedInPage, SurveyCreationPage, TakeSurveyPage, SurveyPage, MySurvey, EditSurvey, SurveyResults що слугуватимуть для відображення змісту WEB-сторінок.

Для авторизації та аутентифікації використовується login та email, при чому є можливість використовувати автентифікацію та логін через Google. Створено модулі, в яких описана логіка для реєстрації, входу та виходу з акаунту.

Для передачі інформації між сервером та клієнтом використовуються HTTP запити. Post запит – відповідає за передачу клієнтом даних на серевер, Get запит – відповідає за отримання клієнтом даних з серверу. Наприклад, запит за допомогою axios.post(url, data, config) виконується функцією та приймає такі аргументи:

- url: рядок, що містить URL, до якого надсилається запит;
- data: об'єкт або рядок, що містять дані, які потрібно надіслати на сервер разом з запитом;
- config: налаштування відповіді від сервера AxiosRequestConfig.

Розглянемо створені класи клієнтської частини WEB-застосунку для проходження опитувань поетапно:

Модуль LoginPage містить методи toggleMode(), ForgotPassword(), handleLogin(e), handleSignUp(e) та атрибути isLogin, name, email, password, message.

Ключовими методами модулю LoginPage є handleLogin(e) та handleSignUp(e).

Розглянемо метод handleLogin(e):

```
const handleLogin = async (e) => {
  e.preventDefault();

  try {
   const response = await axios.post(serverUrl + '/auth/login', {
    email: email,
    password: password
   });

   if (response.data.success) {
    const token = response.data.accessToken;
    if (!token || token === '' || token === undefined) {
     console.error('Login failed: No token received');
     return;
    }
    localStorage.setItem('authToken', token);
    console.log('Login successful');

    navigate('/logged-in');
   } else {
    setMessage('Login failed. Please try again.');
   }
  } catch (error) {
```

```
      console.error('Error:', error);
    }
};
```

Метод handleLogin призначений для обробки події входу користувача. Його основне призначення - виконати аутентифікацію користувача на сервері та обробити результати цієї спроби. Аргумент e - це об'єкт події (event), який передається в метод при його виклику. Цей об'єкт містить інформацію про саму подію, яка відбулася (наприклад, натискання кнопки входу), і дозволяє контролювати її подальшу обробку.

Розглянемо метод handleSignUp(e):

```
const handleSignUp = async (e) => {
  e.preventDefault();
  console.log(e);
  if (!validateEmail(email)) {
    setMessage('Please enter a valid email address.');
    console.log('Invalid email');
    return;
  }

  if (!validatePassword(password)) {
    setMessage('Password must be at least 8 characters long.');
    console.log('Invalid password');
    return;
  }

  try {
    const response = await axios.post(serverUrl + '/auth/signup', {
      name: name,
      email: email,
      password: password,
      picture: 0
```

```
      });
      const resp = response.data;
      if (!resp.success) {
        console.error('Sign up failed');
        if (resp.code === 409) {
          setIsLogin(true);
          setEmail('youremail@gmail.com');
          setPassword('');
          setMessage('User already exists. Please log in.');
        } else {
          setMessage('Sign up failed. Please try again.');
        }
      } else {
        setIsLogin(true);
        setMessage('Sign up successful. You can now log in.');
      }
    } catch (error) {
      console.error('Error:', error);
    }
};
```

Метод handleSignUp використовується для обробки реєстрації користувача. Він перевіряє правильність введених користувачем даних (електронної пошти та пароля), а потім надсилає запит на сервер для реєстрації нового користувача. Аргумент e - це об'єкт події (event), який передається в метод при його виклику. Цей об'єкт містить інформацію про саму подію, яка відбулася, і дозволяє контролювати її подальшу обробку. Метод спочатку відміняє стандартну дію форми (наприклад, відправку даних на сервер без перезавантаження сторінки) за допомогою методу preventDefault(). Потім він перевіряє правильність введених даних, використовуючи допоміжні функції validateEmail і validatePassword. Якщо дані введені неправильно, встановлюється

відповідне повідомлення для користувача. Далі метод надсилає запит на сервер за допомогою бібліотеки Axios. Якщо реєстрація пройшла успішно, повідомлення для користувача встановлюється на "Sign up successful. You can now log in.". Якщо виникла помилка під час реєстрації або користувач вже існує, відповідно обробляються ці помилки і встановлюються відповідні повідомлення для користувача.

Модуль LoggedInPage містить методи createSurvey(), takeSurvey(), mySurveys(), logout(), fetchUserProfile() та атрибут userInfo.

Ключовим методом модулю LoggedInPage є fetchUserProfile().

Розглянемо метод fetchUserProfile():

```
useEffect(() => {
  async function fetchUserProfile() {
    try {
      const response = await fetch(serverUrl + '/user/profile', {
        method: 'GET',
        headers: {
          Authorization: `Bearer ${localStorage.getItem('authToken')}`,
          'Content-Type': 'application/json',
        },
      });
      if (response.ok) {
        const data = await response.json();
        console.log('uerInfo:', data)
        setUserInfo(data);
      } else {
        console.error('Failed to fetch user profile:', response.statusText);
      }
    } catch (error) {
      console.error('Error fetching user profile:', error.message);
    }
  }
```

```
        fetchUserProfile();
}, []);
```

Функція fetchUserProfile() надсилає запит на сервер для отримання профілю користувача. Вона використовує fetch API для виконання запиту та передає токен авторизації в заголовок Authorization. Після успішного отримання відповіді з сервера та перевірки статусу відповіді, дані профілю користувача перетворюються в формат JSON копіюються в userInfo. Якщо відповідь сервера не успішна, відображається повідомлення про помилку у консолі.

Модуль SurveyCreationPage містить методи validateForm(), handleSubmit(), handleAddQuestion() та атрибути questions, quizTitle, quizDescription..

Ключовим методом модулю SurveyCreationPage є handleSubmit().

Розглянемо метод handleSubmit():

```
const handleSubmit = async () => {
    const validation = validateForm();
    if (validation.isValid) {
      const hash = generateSurveyHash(8);
      const authToken = localStorage.getItem('authToken');
      if (authToken) {
        try {
          const response = await axios.post(serverUrl + '/survey/new', {
            questions: questions,
            hash: hash,
            title: quizTitle,
            description: quizDescription,
          }, {
            headers: {
              Authorization: `Bearer ${authToken}`,
            },
          });
```

```
            console.log('Survey submitted successfully:', response.data);
            navigate('/create-survey/success', { state: { hash: hash } });
          } catch (error) {
            console.error('Error submitting survey:', error);
          }
        } else {
          console.error('Auth token not found in localStorage');
          navigate('/')

        }
      } else {
        console.error('Form validation failed');
        alert(validation.message);
      }
  };
```

Метод handleSubmit() викликається при створенні опитування. Спочатку він виконує перевірку форми за допомогою функції validateForm. Якщо валідація успішна, він генерує хеш-код для опитування за допомогою функції generateSurveyHash. Потім він отримує токен авторизації з локального сховища. Якщо токен існує, він виконує запит на сервер за допомогою методу axios.post. У тілі запиту відправляються дані опитування, згенерований хеш, а також заголовок опитування та його опис. Заголовок авторизації додається до заголовка запиту. Після успішного відправлення запиту на сервер і отримання відповіді, виводиться повідомлення про успішне створення опитування в консолі, та користувач перенаправляється на сторінку “/create-survey/success” з передачею хешу опитування у стані. Якщо токен авторизації відсутній, виводиться повідомлення про помилку у консолі, і користувач перенаправляється на домашню сторінку “/”.

Модуль TakeSurveyPage містить метод handleSubmit() та атрибут surveyHash.

Атрибут surveyHash використовується для збереження ідентифікатора опитування.

Метод handleSubmit(event) обробляє подію надсилання форми, перевіряє ідентифікатор опитування та перенаправляє користувача на сторінку з опитуванням.

Модуль SurveyPage містить методи handleEmotionChange(questionId, emotion, value), submitSurvey() та атрибути surveyHash, survey, emotions, isLoading, error, submitte.

Ключовим методом модулю SurveyPage є submitSurvey().

Розглянемо метод submitSurvey():

```
async function submitSurvey() {
    try {
      const authToken = localStorage.getItem('authToken');
      const response = await axios.post(
        `${serverUrl}/survey/submit`,
        {
          surveyHash,
          emotions
        },
        {
          headers: {
            Authorization: `Bearer ${authToken}`
          }
        }
      );

      if (response.data.success) {
        setSubmitted(true);
        navigate('/logged-in');
      }
    } catch (error) {
```

```
    console.error('Error submitting survey:', error);
  }
 }
```

Метод submitSurvey() використовується для відправлення результатів опитування на сервер. Спочатку метод отримує токен авторизації з локального сховища за допомогою localStorage.getItem('authToken'). Потім він виконує запит на сервер за допомогою методу axios.post. У тілі запиту відправляються surveyHash і emotions, які містять ідентифікатор опитування та дані про емоції, зібрані під час опитування.

Модуль MySurvey містить методи fetchData(), copyToClipboard(text), navigateTo(url) та атрибути activeTab, createdSurveys, takenSurveys.

Ключовим методом модулю MySurvey є fetchData().

Розглянемо метод fetchData():

```
const fetchData = async () => {
        try {
          if (activeTab === 'created') {
            const response = await axios.get(`${serverUrl}/survey/created`, {
              headers: {
                Authorization: `Bearer ${authToken}`,
              },
            });
            setCreatedSurveys(response.data);
          } else {
            const response = await axios.get(`${serverUrl}/survey/taken`, {
              headers: {
                Authorization: `Bearer ${authToken}`,
              },
            });
            setTakenSurveys(response.data);
          }
```

```
        } catch (error) {
            console.error("Error fetching surveys:", error);
        }
    };
```

Метод fetchData() використовується для отримання даних опитувань залежно від активної вкладки (“created” або “taken”). Метод визначає, яку вкладку зараз активовано за допомогою змінної activeTab. Якщо активна вкладка “created”, він виконує запит на сервер для отримання створених опитувань користувача. Якщо активна вкладка “taken”, він виконує запит для отримання пройдених опитувань.

Модуль EditSurvey містить методи fetchSurvey(), handleSave(), handleQuestionChange(index, key, value), handleSaveQuestion(questionId, question) та атрибути id, survey, title, description, questions, message, questionMessage.

Ключовим методом модулю EditSurvey є handleSave().

Розглянемо метод handleSave():

```
const handleSave = async () => {
    try {
        const response = await axios.put(`${serverUrl}/survey/${id}`, {
            title,
            description

        }, {
            headers: {
                Authorization: `Bearer ${authToken}`,
            },
        });
        if (response.status === 200) {
            setMessage('Changes saved');
            setTimeout(() => navigate('/my-surveys'), 3000);
        } else {
```

```
            setMessage('Failed to save changes');
          }
        } catch (error) {
          console.error('Error saving survey:', error);
          setMessage('Error saving changes');
        }
  };
```

Метод handleSave() використовується ля оновлення опитування, використовуючи ідентифікатор опитування (id). Дані опитування, такі як назва (title) та опис (description), передаються в тілі запиту. Запит містить заголовок авторизації з токеном, який зберігається в локальному сховищі (localStorage).

Модуль SurveyResults містить методи fetchResults(), fetchAllResults(), prepareChartData(), prepareAverageData(), renderContent(type, content) та атрибути id, data, allData.

Ключовими методами модулю SurveyResults є fetchResults() та fetchAllResults().

Розглянемо метод fetchResults():

```
      const fetchResults = async () => {
        try {
          const response = await axios.get(`${serverUrl}/survey/${id}/results`,
{
            headers: {
              Authorization: `Bearer ${localStorage.getItem('authToken')}`,
            },
          });
          setData(response.data);
        } catch (error) {
          console.error('Error fetching survey results:', error);
        }
    };
```

Метод fetchResults() використовується для отримання результатів опитування. Метод відправляє GET-запит на сервер для отримання результатів опитування, використовуючи ідентифікатор опитування (id). Заголовок запиту містить токен авторизації, який зберігається в локальному сховищі (localStorage). Якщо запит успішний, метод копіює отримані дані в data.

Розглянемо метод fetchAllResults():

```
const fetchAllResults = async () => {
        try {
         const response = await axios.get(`${serverUrl}/survey/${id}/results-all`, {
            headers: {
              Authorization: `Bearer ${localStorage.getItem('authToken')}`,
            },
          });
          setAllData(response.data);
        } catch (error) {
          console.error('Error fetching all survey results:', error);
        }
    };
```

Метод fetchAllResults() використовується для отримання всіх результатів опитування. Метод відправляє GET-запит на сервер для отримання всіх результатів опитування за вказаним ідентифікатором опитування (id). У заголовку запиту включається токен авторизації, який зберігається в локальному сховищі (localStorage). Якщо запит успішний, метод копіює отримані дані в allData.

3.4 Тестування роботи програми та аналіз результатів

Розроблюваний WEB-застосунок було протестовано для підтвердження коректності його роботи.

Було проведено більше 100 запусків додатку, протестовано та оцінено можливості його роботи. Після запуску WEB-застосунку користувач спостерігатиме сторінку логіну/реєстрації (рис. 3.1, 3.2), яка містить в собі функції для створення нового та входу в обліковий запис.

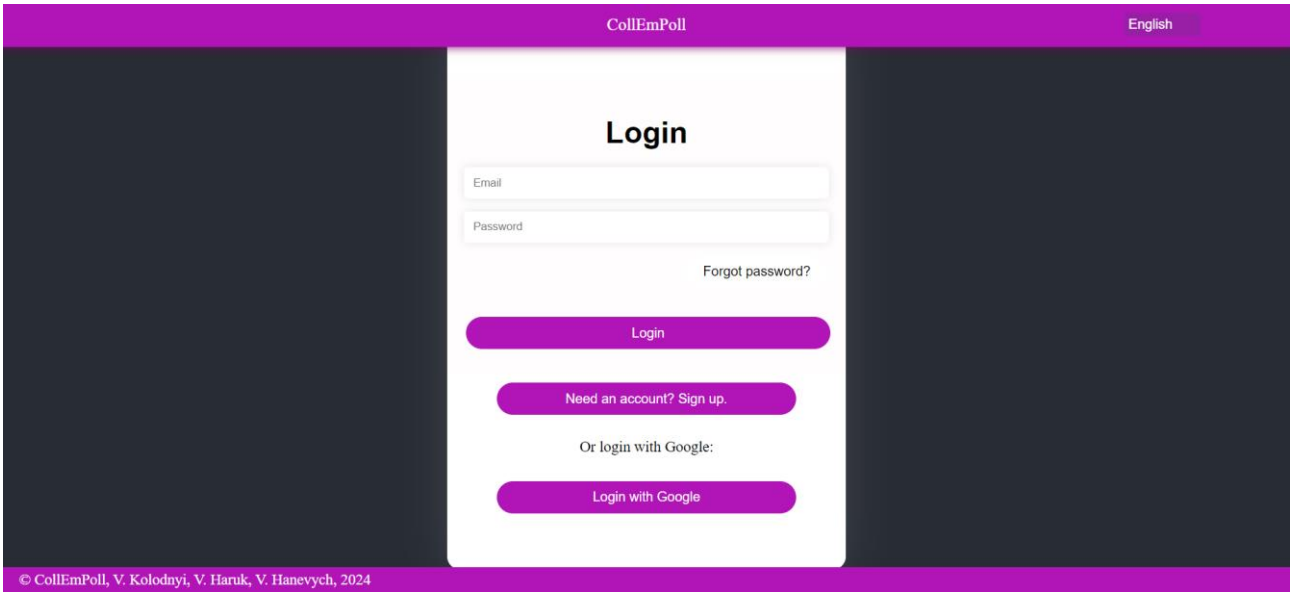

Рисунок 3.1 – Сторінка входу в обліковий запис

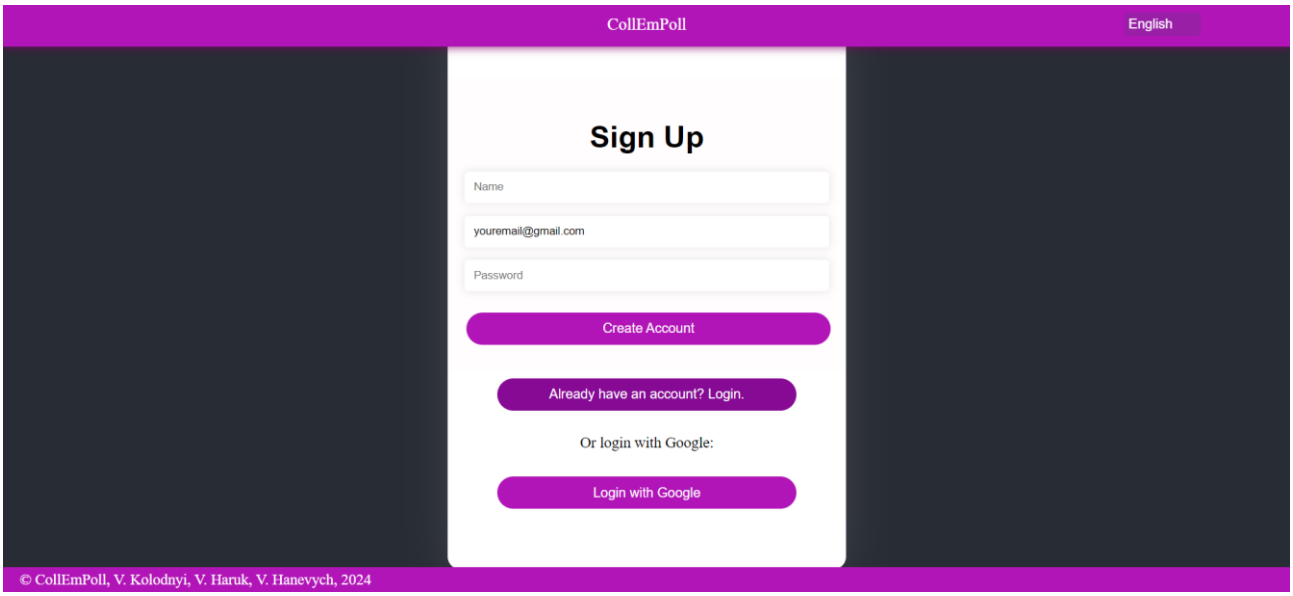

Рисунок 3.2 – Сторінка створення облікового запису

Після успішного входу чи реєстрації користувача його особистий кабінет (рис. 3.3), в якому користувач може створити опитування (рис. 3.4), долучитись до опитування (рис. 3.5) та переглянути власні пройдені та створені опитування (рис. 3.6).

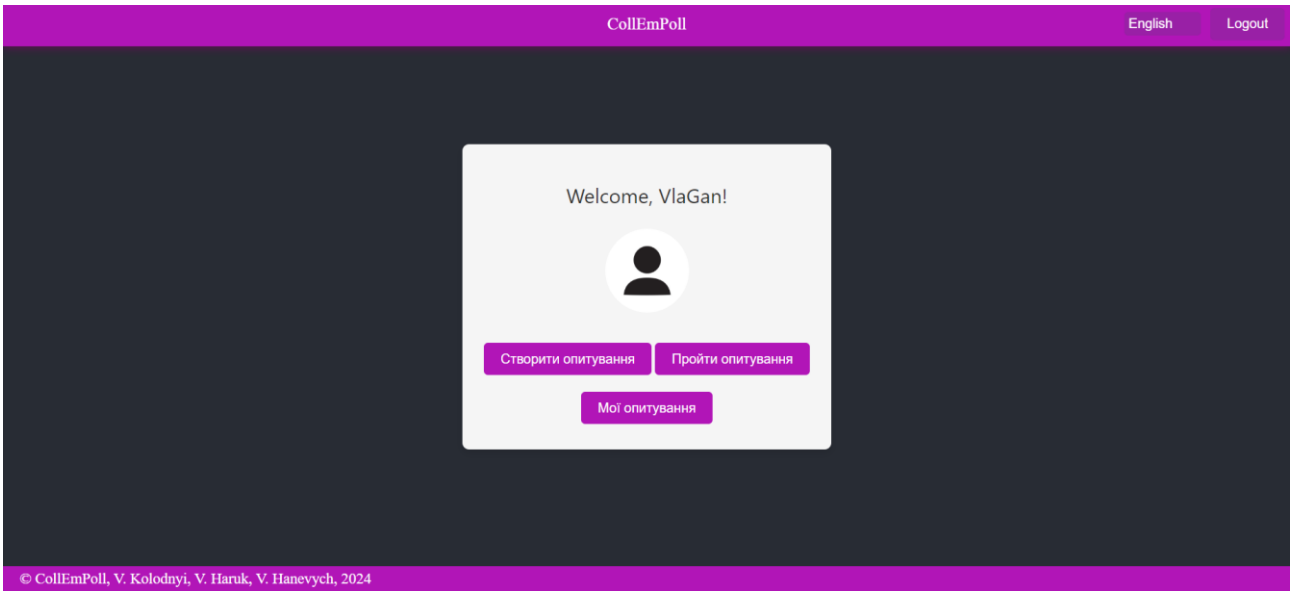

Рисунок 3.3 – Особистий кабінет користувача

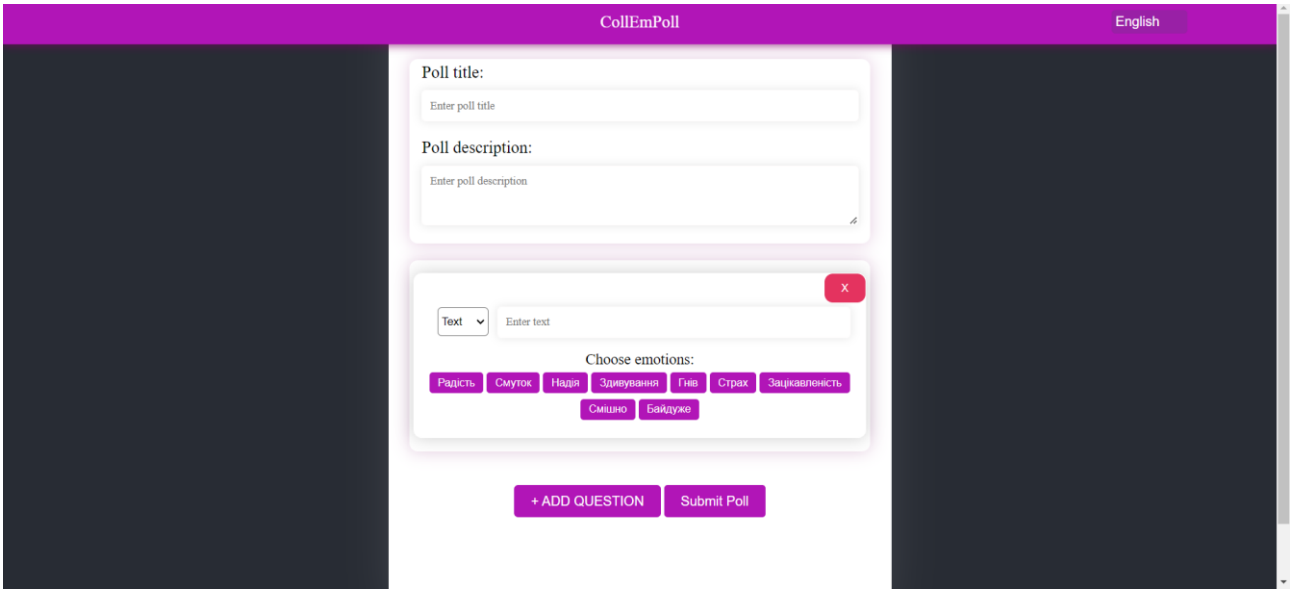

Рисунок 3.4 – Сторінка створення опитування

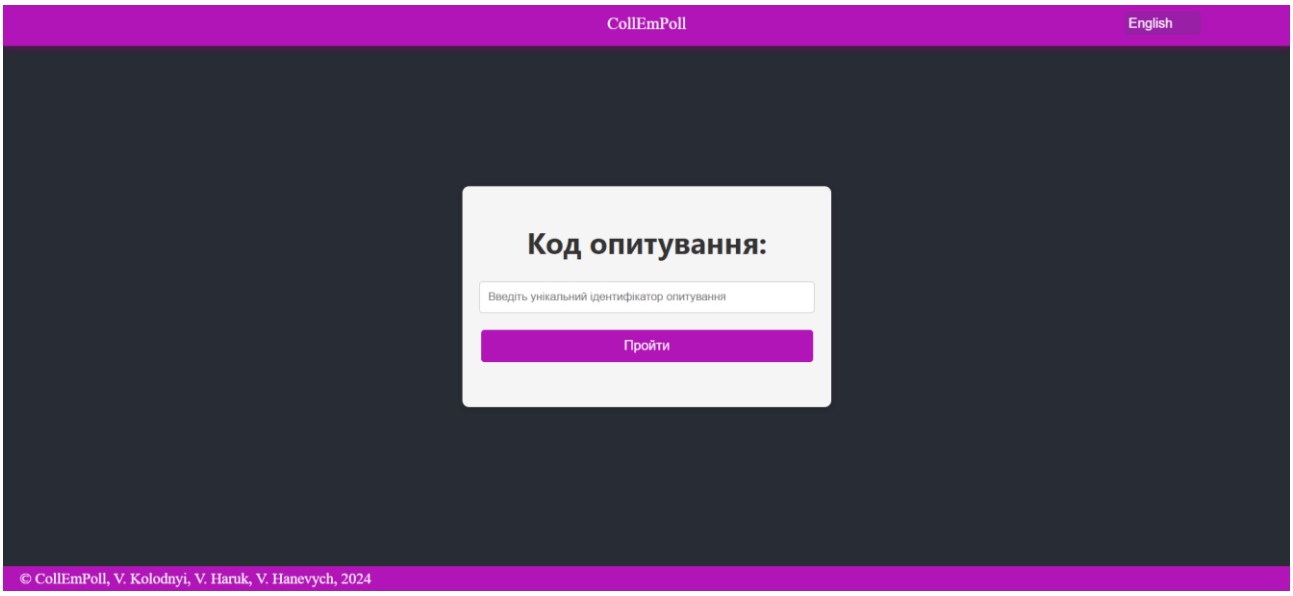

Рисунок 3.5 – Сторінка доєднання до опитування

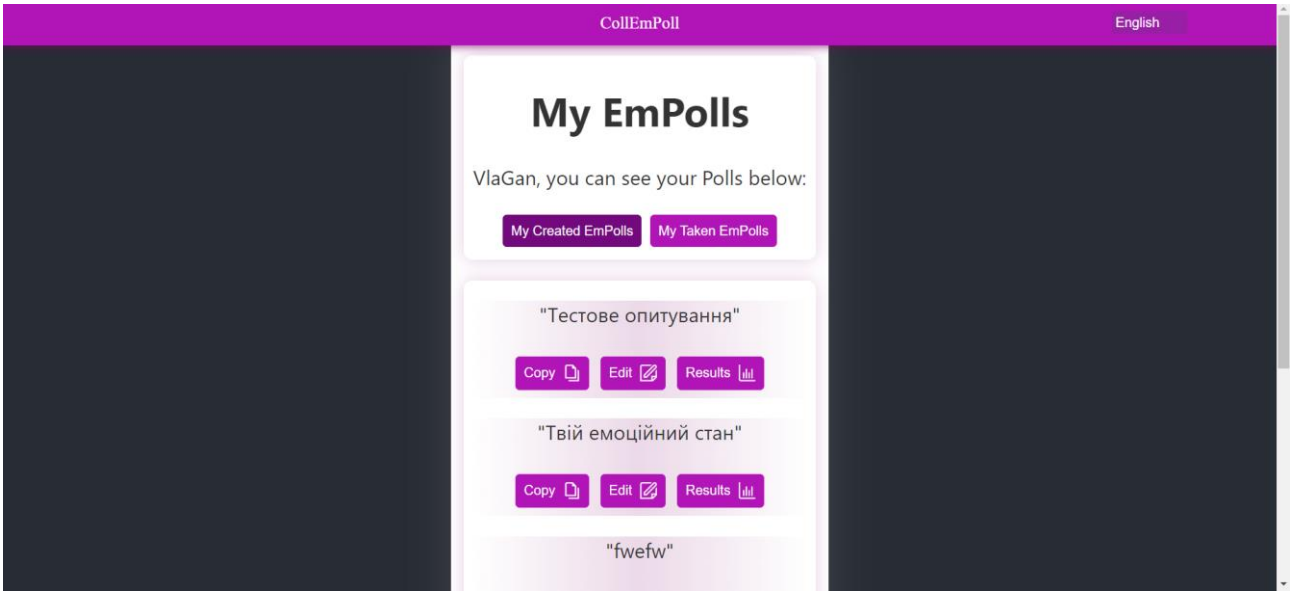

Рисунок 3.6 – Сторінка створених та пройдених опитувань користувача

Для того, щоб долучитись до опитування, користувачеві необхідно знати унікальний код опитування, після чого перейти на сторінку доєднання до опитування (рис. 3.5), ввести код у поле вводу та натиснути кнопку підтвердження, тоді користувача перенаправить на сторінку проходження опитування (рис 3.7).

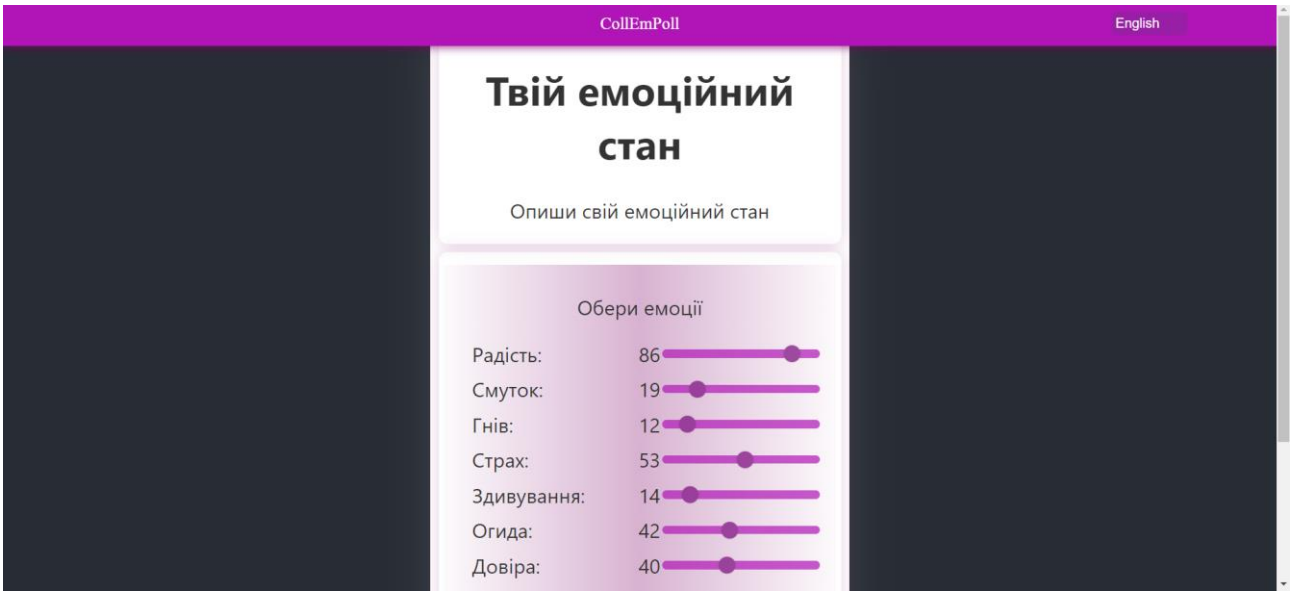

Рисунок 3.7 – Сторінка проходження опитування

На рис 3.6 зображено лише створені користувачем опитування, для того, щоб переглянути пройдені, слід натиснути кнопку пройдених опитувань, після чого відобразяться пройдені користувачем опитування (рис. 3.8). Для редагування опитування слід перейти у створені опитування і натиснути кнопку редагування біля потрібного (рис. 3.9). Для перегляду результатів слід натиснути кнопку результатів біля потрібного опитування (рис. 3.10).

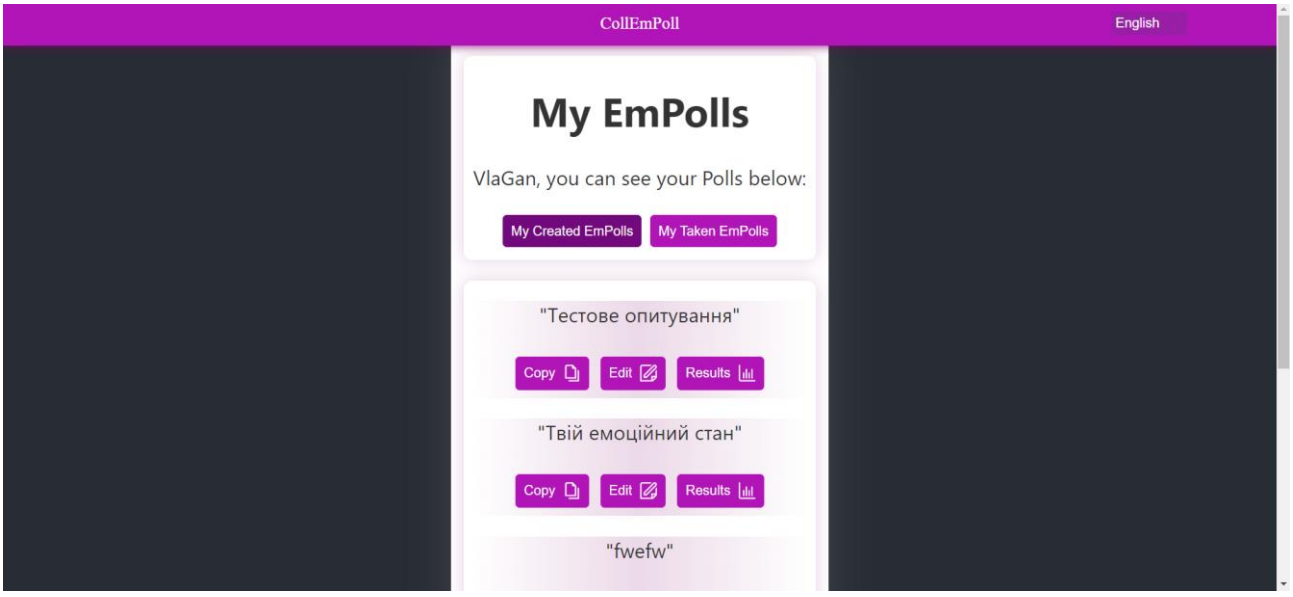

Рисунок 3.8 – Пройдені користувачем опитування

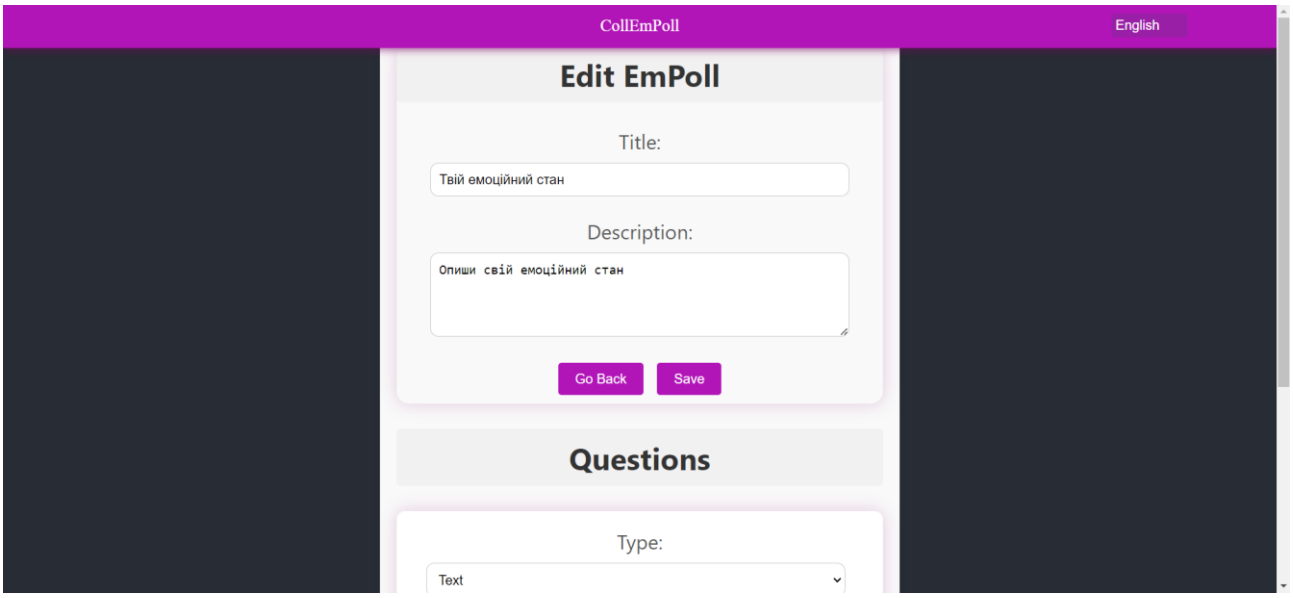

Рисунок 3.9 – Редагування створених користувачем опитувань

На рисунку 3.9 зображено функціонал для редагування опитувань, створених користувачем. В процесі редагування потрібного опитування, користувач може змінити назву й опис опитування та редагувати питання опитування, маючи змогу змінити їх тип (текст, фото, відео) та спектр емоцій для їх оцінки. Для збереження опитування користувачеві потрібно натиснути кнопку збереження опитування, яка розташована під його описом. Для повернення до опитувань користувача без змін, потрібно натиснути кнопку повернення до опитувань, яка розташована біля кнопки збереження.

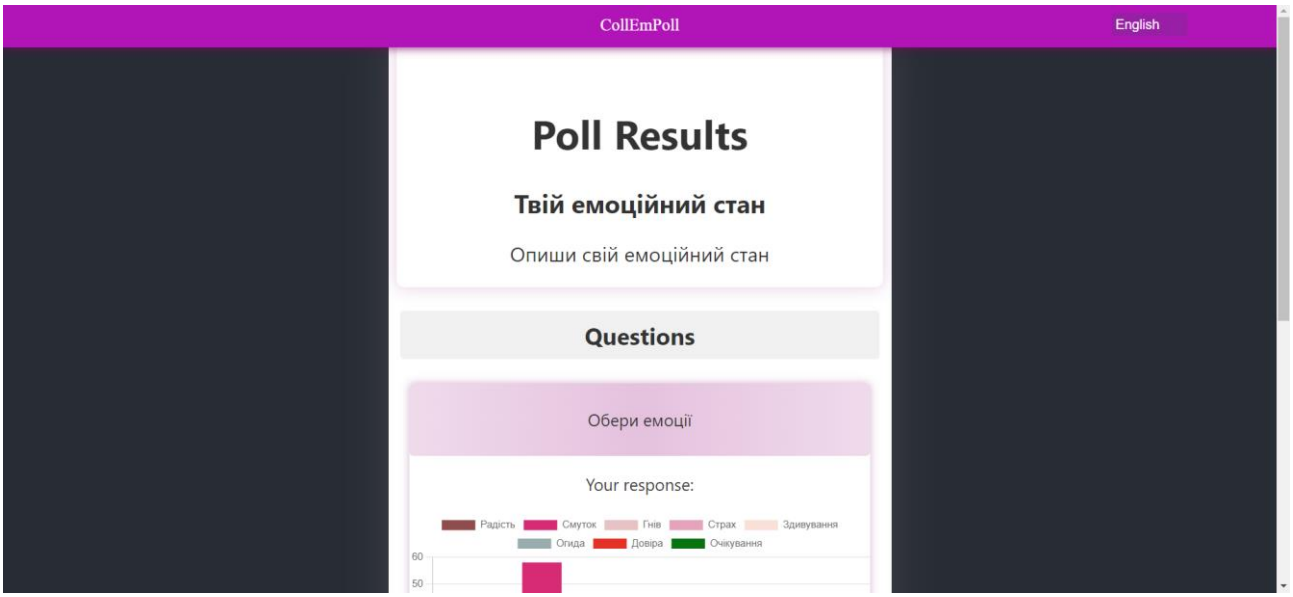

Рисунок 3.10 – Результати опитування

Рисунок 3.10 – Продовження

В табл. 3.3 подано порівняння результатів тестування розробленого WEB-застосунку з аналогами.

В результаті розробки було досягнено поставленої мети за рахунок реалізації функцій, що відсутні в програмах-аналогах, а саме:

- реалізовано новий формат подачі відповідей на запитання опитувань у вигляді унікального спектру емоцій;
- реєстрація за допомогою email;
- Google авторизація;
- кастомізований формат подачі результатів, який у даному випадку полягає у вигляді оцінки рівню емоцій з допомогою повзунків;

Таблиця 3.3 – Результати тестування розробленого WEB-застосунку порівняно з аналогами

Додаток	Спектр емоцій	Реєстрація за допомогою email	Google авторизація	Кастомізований формат подачі результатів
Google Forms	-	+	+	-
Typeform	-	+	+	+
Qualtrics	-	+	+	-
Розроблений застосунок	+	+	+	+

Отже, проаналізувавши та протестувавши розроблений додаток можна стверджувати, що поставлених цілей досягнуто. За результатами порівняння табл. 3.3 можна побачити, що розроблений програмний засіб має покращений функціонал, принаймні на декілька пунктів у порівнянні з програмами-аналогами. Розроблений WEB-застосунок для проходження опитувань має ширший функціонал за рахунок нового формату подачі відповідей на запитання опитувань у вигляді спектру емоцій.

3.5 Висновок

В цьому розділі проведено аналіз найчастіше використовуваних для розробки WEB-застосунків мов програмування: JavaScript, C#, Python. Визначено особливості, переваги та недоліки кожної з них, а також доцільність у використанні для поставленого завдання. В результаті аналізу для реалізації програми було обрано мову програмування JavaScript. Ключовими перевагами, які відіграли головну роль у виборі є: статистична типізація, що дозволяє зменшити кількість можливих помилок при обрані типу; явна типізація дозволить вказувати типи змінних, щоб уникнути можливих помилок.

Розроблено UML-діаграму класів WEB-застосунку для проведення опитувань. На її основі програмно реалізовано WEB-застосунок.

Проведено тестування розробленого ресурсу та порівняння з аналогами. З результатів порівняння зроблено висновок, що основні функції, які мали бути розробленими – реалізовано.

Розроблений програмний засіб має покращений функціонал, принаймні на декілька можливостей у порівнянні з програмами-аналогами, а саме: реєстрація за допомогою email чи Google, можливість редагувати опитування. Крім того, реалізовано новий формат подачі відповідей на запитання опитувань у вигляді унікального спектру емоцій.

ВИСНОВКИ

Всі задачі, поставлені для реалізації WEB-застосунку для проходження опитувань були виконані в повному обсязі, а саме: обґрунтовано доцільність розробки WEB-застосунку для проходження опитувань; спроектовано WEB-застосунок для проходження опитувань; програмно реалізовано клієнтську частину WEB-застосунку для проходження опитувань; виконано тестування програми та проведено аналіз отриманих результатів; розроблено інструкцію користувача.

У бакалаврській дипломній роботі було створено WEB-застосунок для проходження опитувань. Досліджено існуючі платформи для реалізації застосунку для проходження опитувань та обрано найбільш прийнятну – WEB, через її захищеність та універсальність.

В роботі представлено загальну UML-діаграму класів та описано їх методи, на основі якої відбулася подальша розробка. Розроблено структуру принципу роботи системи та схему алгоритму функціонування WEB-застосунку. Схема алгоритму допомогла зрозуміти, як покроково має працювати програма. Крім того, розроблено структуру WEB-застосунку для проходження опитувань. В результаті, реалізовано WEB-застосунок для проходження опитувань. Додаток розроблений мовою програмування JavaScript з використанням фреймворку React.

Розроблений WEB-застосунок успішно пройшов тестування та у повній мірі відповідає заданим вимогам.

Мета роботи, яка полягала в розширенні функціональних можливостей WEB-застосунку для проходження опитувань, досягнута за рахунок того, що у порівнянні з аналогами, введено декілька нових можливостей, а саме: реєстрація за допомогою email чи Google, можливість редагувати опитування. Крім того, реалізовано новий формат подачі відповідей на запитання опитувань у вигляді унікального спектру емоцій, що дозволяє полегшити вираження та аналіз емоцій респондентів, які користуються WEB-застосунком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ганевич В. М., Гарук В.В., Колодний В.В. Аналіз передумов розробки WEB-застосунку для проходження опитувань. *LIII Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації.* URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20883 (дата звернення: 06.05.2024).
2. Онлайн опитування. URL: https://cpd.com.ua/uk/online-opytuvannya/ (дата звернення: 06.05.2024).
3. Що таке Веб-додаток. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 06.05.2024).
4. Google Forms. URL: https://www.template.net/google/google-form/ (дата звернення: 07.05.2024).
5. What is SurveyMonkey? URL: https://www.genroe.com/blog/what-is-surveymonkey/7979 (дата звернення: 07.05.2024).
6. Typeform: People-Friendly Forms and Surveys. URL: https://www.typeform.com/ (дата звернення: 23.05.2023).
7. Qualtrics. URL: https://isa.ncsu.edu/surveys/qualtrics/what-is-qualtrics/ (дата звернення: 08.05.2024).
8. Клієнт-серверна архітектура. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 12.05.2024).
9. How to Use Two-Factor Authentication? URL: https://docs.vultr.com/how-to-use-two-factor-authentication-with-sudo-and-ssh-on-linux-with-google-authenticator (дата звернення: 12.05.2024).
10. Що таке ООП простими словами? URL: https://foxminded.ua/shcho-take-oop/ (дата звернення: 12.05.2024).
11. Python. URL: https://www.python.org/ (дата звернення: 15.05.2024).
12. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 15.05.2024).

13. C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 15.05.2024).
14. Django. URL: https://www.djangoproject.com/ (дата звернення: 17.05.2024).
15. Flask. URL: https://flask.palletsprojects.com/en/3.0.x/tutorial/ (дата звернення: 17.05.2024).
16. Node.js з Express. URL: https://expressjs.com/ (дата звернення: 21.05.2024).
17. React. URL: https://react.dev/ (дата звернення: 21.05.2024).
18. Visual Studio: IDE and Code Editor for Software Developers and Teams (microsoft.com). URL: https://visualstudio.microsoft.com/ (дата звернення: 25.05.2024).
19. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення: 25.05.2024).
20. Rider: The Cross-Platform. NET IDE from JetBrains. URL: https://www.jetbrains.com/rider/ (дата звернення: 25.05.2024).

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: WEB-застосунок для проходження опитувань. Частина 2. Клієнтська частина

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 91,54% Схожість 8,46%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ганевич В. М.

Керівник роботи _____ Колодний В. В.

ДОДАТОК Б

Лістинг програми

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};
export default reportWebVitals;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
reportWebVitals();

function App() {
  const authToken = localStorage.getItem('authToken');
  return (
    <Router>
      <script src="http://localhost:8097"></script>
      <div className="App">
        <header className="App-header">
          <Routes>
            <Route path="/" element={authToken ? <Navigate replace to="/logged-in" />
: <Navigate replace to="/login" />} />
            <Route path="/logged-in" element={<LoggedInPage />} />
            <Route path="/login" element={<LoginPage />} />
            <Route path="/survey/:surveyHash" element={<SurveyPage />} />
            <Route path="/storeToken" element={<StoreToken />} />
            <Route path="/take-survey" element={<TakeSurveyPage />} />
            <Route path="/create-survey" element={<SurveyCreationPage />} />
            <Route path="/create-survey/success" element={<HashPage />} />
            <Route path="/my-surveys" element={<MySurvey />} />
            <Route path="/edit-survey/:id" element={<EditSurvey />} />
            <Route path="/survey-results/:id" element={<SurveyResults />} />
```

```
          <Route path="/forgot-password" element={<ForgotPasswordPage />} />
          <Route                                    path="/survey-results-admin/:survey_id"
element={<SurveyPassedUsersAdmin />} />
          <Route   path="/results-admin/:id/:user_id"   element={<SurveyResultsAdmin
/>} />

        </Routes>
       </header>
     </div>
   </Router>
  );
}
export default App;

function LoginPage() {
  const [isLogin, setIsLogin] = useState(true);
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');

  let navigate = useNavigate();

  const toggleMode = () => {
    setIsLogin(!isLogin);
    setEmail('youremail@gmail.com');
    setPassword('');
  };

  const handleLogin = async (e) => {
    e.preventDefault();

    try {
      const response = await axios.post(serverUrl + '/auth/login', {
        email: email,
        password: password
      });

      if (response.data.success) {
        const token = response.data.accessToken;
        if (!token || token === '' || token === undefined) {
          console.error('Login failed: No token received');
          return;
        }
        localStorage.setItem('authToken', token);
```

```
        console.log('Login successful');

        navigate('/logged-in');
       } else {
        setMessage('Login failed. Please try again.');
       }
     } catch (error) {
      console.error('Error:', error);
     }
   };

const handleSignUp = async (e) => {
  e.preventDefault();
  console.log(e);
  if (!validateEmail(email)) {
    setMessage('Please enter a valid email address.');
    console.log('Invalid email');
    return;
  }

  if (!validatePassword(password)) {
    setMessage('Password must be at least 8 characters long.');
    console.log('Invalid password');
    return;
  }

  try {
    const response = await axios.post(serverUrl + '/auth/signup', {
      name: name,
      email: email,
      password: password,
      picture: 0
    });
    const resp = response.data;
    if (!resp.success) {
      console.error('Sign up failed');
      if (resp.code === 409) {
        setIsLogin(true);
        setEmail('youremail@gmail.com');
        setPassword(''); // Clear the password field
        setMessage('User already exists. Please log in.');
      } else {
        setMessage('Sign up failed. Please try again.');
      }
    } else {
      setIsLogin(true);
```

```
      setMessage('Sign up successful. You can now log in.');
    }
  } catch (error) {
    console.error('Error:', error);
  }
};

const handleForgotPassword = () => {
  navigate('/forgot-password');
};

return (

  <div className="LoginPage">
    <TopPanelBase/>
    {message && <p>{message}</p>}
    <div className="login-container">
    <form onSubmit={isLogin ? handleLogin : handleSignUp} className="login-
form">
    <h2>{isLogin ? 'Login' : 'Sign Up'}</h2>

      {!isLogin && (
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
          placeholder="Name" required/>
      )}
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Email" required />
      <input
        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        placeholder="Password" required/>
          {isLogin && (
            <div className="forgot-password-container">
              <button type="button" onClick={handleForgotPassword} className='login-
forgot-password'>
                Forgot password?
              </button>
            </div>
          )}
```

```
      <button type="submit" className="login-submitbtn">{isLogin ? 'Login' : 'Create
Account'}</button>
    </form>
    </div>

    <button onClick={toggleMode} className="login-register-swbtn">
      {isLogin ? 'Need an account? Sign up.' : 'Already have an account? Login.'}
    </button>
    <p className='login-google-text'>Or login with Google:</p>
    <GoogleAuthButton/>
    <BotPanel is_fixed={true}/>
  </div>
);
}
export default LoginPage;
function LoggedInPage() {
  const [userInfo, setUserInfo] = useState(null);
  let navigate = useNavigate();

  useEffect(() => {
    async function fetchUserProfile() {
      try {
        const response = await fetch(serverUrl + '/user/profile', {
          method: 'GET',
          headers: {
            Authorization: `Bearer ${localStorage.getItem('authToken')}`,
            'Content-Type': 'application/json',
          },
        });
        if (response.ok) {
          const data = await response.json();
          console.log('uerInfo:', data)
          setUserInfo(data);
        } else {
          console.error('Failed to fetch user profile:', response.statusText);
        }
      } catch (error) {
        console.error('Error fetching user profile:', error.message);
      }
    }
    fetchUserProfile();
  }, []);

  const createSurvey = () => {
    console.log("Create a new survey");
    navigate('/create-survey');
```

```
  };

  const takeSurvey = () => {
    console.log("Take a survey");
    navigate('/take-survey');
  };

  const mySurveys = () => {
    console.log("View my surveys");
    navigate('/my-surveys');
  };

  console.log(userInfo);
  const logout = async () => {
    localStorage.removeItem('authToken');
    window.location.href = '/';
  };

  return (
    <div className="LoggedInPage">
      <TopPanelBase/>

      {userInfo ? (
        <div>
          <p>Welcome, {userInfo.name}!</p>
          {userInfo.picture && userInfo.picture.startsWith('http') ? (
            <img src={userInfo.picture} alt="Profile" />
          ) : (
            // If picture URL doesn't contain http, render a default image
            <img
src="https://t3.ftcdn.net/jpg/05/87/76/66/360_F_587766653_PkBNyGx7mQh9l1XXPtCAq1lBgOsLl6xH.jpg" alt="Profile" />
          )}
          <div>
            <button onClick={logout} className="logout-button">Logout</button>
          </div>
          <div>
            <button type="button" onClick={createSurvey}>Створити опитування</button>
            <button type="button" onClick={takeSurvey}>Пройти опитування</button>
            <button type="button" onClick={mySurveys}>Мої опитування</button>
          </div>
        </div>
      ) : (

        <div>
```

```
        <p>No user information available. Please Log In!</p>
        <div>
          <button onClick={logout} className="logout-button">Logout</button>
        </div>
        <div>
          <button type="button" onClick={logout}>Login</button>
        </div>

      </div>
    )}
    <BotPanel is_fixed={true}/>
  </div>
 );
}
export default LoggedInPage;

const ForgotPasswordPage = () => {
 const [email, setEmail] = useState('');
 const [message, setMessage] = useState('');
 let navigate = useNavigate();

 const handleForgotPassword = async (e) => {
  e.preventDefault();
  try {
   const response = await axios.post(serverUrl + '/auth/forgot-password', {
    email: email,
   },{
    Authorization: `Bearer ${localStorage.getItem('authToken')}`,
    'Content-Type': 'application/json',
   });

   if (response.data.success) {
    setMessage('A password reset link has been sent to your email.');
   } else {
    setMessage('Failed to send passwor. Please try again or contact admin
CollEmPoll@gmail.com');
   }
  } catch (error) {
   console.error('Error:', error);
   setMessage('An error occurred. Please try again.');
  }
 };

 return (
  <div className="forgot-pssw-main">
   <TopPanelBase />
```

```
      <div className='forgot-pssw-top'>
        <h2>Forgot Password?</h2>
        <p>If you forgot account password, you can reset it using email:</p>
      </div>
      <div className='forgot-password-form-container'>
        <form onSubmit={handleForgotPassword} className="forgot-password-
form">
          {message && <p style={{fontSize: '20px'}}>{message}</p>}
          <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            placeholder="Email" required/>
          <button type="submit" className="login-submitbtn">
            Send Reset Link
          </button>
        </form>
      </div>
      <p className='forgot-pssw-bottom-p'>All are ok? Return to login:</p>
      <button onClick={() => navigate("/login")} className="login-register-
swbtn">Login</button>
      <BotPanel is_fixed={true}/>
    </div>
  );
};
export default ForgotPasswordPage;

function generateSurveyHash(length) {
  const characters =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345678
9';
  const charactersLength = characters.length;
  let hash = '';
  for (let i = 0; i < length; i++) {
    hash += characters.charAt(Math.floor(Math.random() * charactersLength));
  }
  return hash;
}

export function SurveyCreationPage() {
  let navigate = useNavigate();
  const [questions, setQuestions] = useState([{ type: "text", answer: "", emotions:
defaultEmotions }]);
  const [quizTitle, setQuizTitle] = useState('');
  const [quizDescription, setQuizDescription] = useState('');
```

```
  const validateForm = () => {
    for (let question of questions) {
      if (question.type === 'text' && question.answer.trim() === "") {
        return { isValid: false, message: "Please fill in all text questions before submitting." };
      } else if ((question.type === 'image' || question.type === 'video') && !question.answer.startsWith('http')) {
        return { isValid: false, message: `Please enter a valid ${question.type} URL starting with http or https.` };
      } else if (question.emotions.length === 0) {
        return { isValid: false, message: "Please select at least one emotion for each question." };
      }
    }
    return { isValid: true, message: "" };
  };

  const handleSubmit = async () => {
    const validation = validateForm();
    if (validation.isValid) {
      const hash = generateSurveyHash(8);
      const authToken = localStorage.getItem('authToken');
      if (authToken) {
        try {
          const response = await axios.post(serverUrl + '/survey/new', {
            questions: questions,
            hash: hash,
            title: quizTitle,
            description: quizDescription,
          }, {
            headers: {
              Authorization: `Bearer ${authToken}`, // Include bearer token in the headers
            },
          });
          console.log('Survey submitted successfully:', response.data);
          navigate('/create-survey/success', { state: { hash: hash } });
        } catch (error) {
          console.error('Error submitting survey:', error);
        }
      } else {
        console.error('Auth token not found in localStorage');
        navigate('/')

      }
    } else {
      console.error('Form validation failed');
```

```
    alert(validation.message);
  }
};

const handleAddQuestion = () => {
  setQuestions([...questions, { answer: "", type: "text", emotions: defaultEmotions }]);
};

return (
  <div className='base-unfixed-footer'>
  <div className='base-unfixed-footer-content'>
  <div className='CreateSyrveyPage'>
  <div className="survey-creation-page">
    <div className='syrvey-top-container'> <div className='syrvey-top'>
    <div className='syrvey-top-text'><a>Poll title:</a></div>
    <input
      type="text"
      placeholder="Enter poll title"
      value={quizTitle}
      onChange={(e) => setQuizTitle(e.target.value)}/>

    <div className='syrvey-top-text'><a>Poll description:</a></div>
    <textarea
      placeholder="Enter poll description"
      value={quizDescription}
      onChange={(e) => setQuizDescription(e.target.value)}></textarea>
    </div></div>
    <div className='syrvey-questions-container'>
    <div className='syrvey-questions'>
    <QuestionList
     questions={questions}
     setQuestions={setQuestions}/>
    </div>
    </div>
    <div className="syrvey-add-submit-container">
      <button onClick={handleAddQuestion} className="add-question-button">
        + ADD QUESTION
      </button>

      <button onClick={handleSubmit} className="submit-survey-button">
        Submit Poll
      </button>
    </div>
  </div>
  </div></div>
  <TopPanelBase/>
```

```
    <BotPanel is_fixed={false}/>
    </div>
  );}

function HashPage() {
  const location = useLocation();
  const { hash } = location.state || {};
  let navigate = useNavigate();

  const copyToClipboard = (text) => {
    if (navigator.clipboard && window.isSecureContext) {
        navigator.clipboard.writeText(text).then(() => {
            alert('Link copied to clipboard: ' + text);
        }).catch((err) => {
            console.error('Failed to copy: ', err);
        });
    } else {
        try {
            let textarea = document.createElement("textarea");
            textarea.value = text;
            document.body.appendChild(textarea);
            textarea.focus();
            textarea.select();
            let successful = document.execCommand('copy');
            if (successful) {
                alert('Link copied to clipboard: ' + text);
            } else {
                throw new Error('Copy command was unsuccessful');
            }
            document.body.removeChild(textarea);
        } catch (err) {
            console.error('Fallback: Oops, unable to copy', err);
        }
    }
  };

  return (
    <div className='base-unfixed-footer'>
    <div className='base-unfixed-footer-content'>
    <div className='hash-page'>
      <h1 className='hash-page_main_text'>Poll created successfully!</h1>
      <div className='hash-page-main'>
        <div className='hash-page-unit'>
          <p>Poll Code: {hash ? hash : 'No hash provided'}</p>
          <div className='base-button'>
```

```
          <button          className='base-iconed-button'          onClick={()          =>
copyToClipboard(hash ? hash : '')}>
            Copy code<img src={copy_icon}  alt='copy-poll-hash' className="icon"/>
          </button>
        </div>
       </div>

       <div className='hash-page-unit'>
        <p>Poll                                    Link:                                    <a
href={`${reactUrl}/survey/${hash}`}>{`${reactUrl}/survey/${hash}`}</a></p>
        <div className='base-button'>
          <button          className='base-iconed-button'          onClick={()          =>
copyToClipboard(`${reactUrl}/survey/${hash}`)}>
            Copy link<img src={copy_icon}  alt='copy-poll-link' className="icon"/>
          </button>
        </div>
       </div>

       <div className='hash-page-unit'>
        <p>You can see your Polls here:</p>
        <div className='base-button'>
          <button onClick={() => navigate("/my-surveys")}>My Polls</button>
        </div>
       </div>
     </div>
    </div>
    </div>

    <TopPanelBase/>
    <BotPanel is_fixed={false}/>
    </div>
  );
}
export default HashPage;

function EditSurvey() {
    const { id } = useParams();
    const [survey, setSurvey] = useState(null);
    const [title, setTitle] = useState('');
    const [description, setDescription] = useState('');
    const [questions, setQuestions] = useState([]);
    const [message, setMessage] = useState('');
    const [questionMessage, setQuestionMessage] = useState('');
    const navigate = useNavigate();
    const authToken = localStorage.getItem('authToken');
```

```
useEffect(() => {
    const fetchSurvey = async () => {
        try {
            const response = await axios.get(`${serverUrl}/survey/${id}`, {
                headers: {
                    Authorization: `Bearer ${authToken}`,
                },
            });
            setSurvey(response.data);
            setTitle(response.data.title);
            setDescription(response.data.description);
            setQuestions(response.data.questions);
        } catch (error) {
            console.error('Error fetching survey:', error);
        }
    };
    fetchSurvey();
}, [id, authToken]);

const handleSave = async () => {
    try {
        const response = await axios.put(`${serverUrl}/survey/${id}`, {
            title,
            description

        }, {
            headers: {
                Authorization: `Bearer ${authToken}`,
            },
        });
        if (response.status === 200) {
            setMessage('Changes saved');
            setTimeout(() => navigate('/my-surveys'), 3000);
        } else {
            setMessage('Failed to save changes');
        }
    } catch (error) {
        console.error('Error saving survey:', error);
        setMessage('Error saving changes');
    }
};

const handleQuestionChange = (index, key, value) => {
    const newQuestions = [...questions];
    newQuestions[index][key] = value;
    setQuestions(newQuestions);
```

```
    };

    const handleSaveQuestion = async (questionId, question) => {
        try {
            const response = await axios.put(`${serverUrl}/survey/${id}/questions/${questionId}`, question, {
                headers: {
                    Authorization: `Bearer ${authToken}`,
                },
            });
            if (response.status === 200) {
                setQuestionMessage('Question changes saved');
            } else {
                setQuestionMessage('Failed to save question changes');
            }
        } catch (error) {
            console.error('Error saving question:', error);
            setQuestionMessage('Error saving question changes');
        }
    };
    useEffect(() => {
        if (message) {
            const timer = setTimeout(() => setMessage(''), 3000);
            return () => clearTimeout(timer);
        }
    }, [message]);
    useEffect(() => {
        if (questionMessage) {
            const timer = setTimeout(() => setQuestionMessage(''), 3000);
            return () => clearTimeout(timer);
        }
    }, [questionMessage]);

    if (!survey) {
        return <div>Loading...</div>;
    }
    return (
        <div className='base-unfixed-footer'>
        <div className='base-unfixed-footer-content'>
        <div className="edit-survey-page">
            <TopPanelBase/>
            <div className="edit-survey-container">
                <div className='edit-survey-top'>
                <h2>Edit EmPoll</h2>
                <div>
                    <label>Title:</label>
```

```
            <input
              type="text"
              value={title}
              onChange={(e) => setTitle(e.target.value)}/>
          </div>
          <div>
            <label>Description:</label>
            <textarea
              value={description}
              onChange={(e) => setDescription(e.target.value)}></textarea>
          </div>
          <button onClick={() => navigate('/my-surveys')}>Go Back</button>
          <button onClick={handleSave}>Save</button>

          </div>
          {message && <p className="message">{message}</p>}
          <h2>Questions</h2>
          {questions.map((question, index) => (
            <div key={question.question_id} className="question-item">
              <div>
                <label>Type:</label>
                <select
                  value={question.type}
                  onChange={(e) => handleQuestionChange(index, 'type',
e.target.value)}>
                  <option value="text">Text</option>
                  <option value="image">Image</option>
                  <option value="video">Video</option>
                </select>
              </div>
              <div>
                <label>Question:</label>
                <input
                  type="text"
                  value={question.question}
                  onChange={(e) => handleQuestionChange(index, 'question',
e.target.value)}/>
              </div>
              <div>
                <label>Emotions:</label>
                <input
                  type="text"
                  value={question.emotions.join(', ')}
                  onChange={(e) => handleQuestionChange(index, 'emotions',
e.target.value.split(', '))}/>
              </div>
```

```
                <button onClick={() => handleSaveQuestion(question.question_id, question)}>Save Question</button>
            </div>))}
        {questionMessage && <p className="question-message">{questionMessage}</p>}
      </div>
    </div>
    </div>
    <BotPanel is_fixed={false}/>
    </div>
  );
}
export default EditSurvey;

function MySurvey() {
  const [activeTab, setActiveTab] = useState('created');
  const [createdSurveys, setCreatedSurveys] = useState([]);
  const [takenSurveys, setTakenSurveys] = useState([]);
  const [userInfo, setUserInfo] = useState(null);
  let navigate = useNavigate();
  const authToken = localStorage.getItem('authToken');

  useEffect(() => {
    const fetchData = async () => {
      try {
        if (activeTab === 'created') {
          const response = await axios.get(`${serverUrl}/survey/created`, {
            headers: {
              Authorization: `Bearer ${authToken}`,
            },
          });
          setCreatedSurveys(response.data);
        } else {
          const response = await axios.get(`${serverUrl}/survey/taken`, {
            headers: {
              Authorization: `Bearer ${authToken}`,
            },
          });
          setTakenSurveys(response.data);
        }
      } catch (error) {
        console.error("Error fetching surveys:", error);
      }
    };

    async function fetchUserProfile() {
```

```
      try {
        const response = await fetch(serverUrl + '/user/profile', {
          method: 'GET',
          headers: {
            Authorization: `Bearer ${localStorage.getItem('authToken')}`,
            'Content-Type': 'application/json',
          },
        });
        if (response.ok) {
          const data = await response.json();
          console.log('uerInfo:', data)
          setUserInfo(data);
        } else {
          console.error('Failed to fetch user profile:', response.statusText);
        }
      } catch (error) {
        console.error('Error fetching user profile:', error.message);
      }
    };

    fetchData();
    fetchUserProfile();
  }, [activeTab, authToken]);

  const copyToClipboard = (text) => {
    if (navigator.clipboard && window.isSecureContext) {
      navigator.clipboard.writeText(text).then(() => {
        alert('Link copied to clipboard! ' + text);
      }).catch((err) => {
        console.error('Failed to copy: ', err);
      });
    } else {
      try {
        let textarea = document.createElement("textarea");
        textarea.value = text;
        document.body.appendChild(textarea);
        textarea.focus();
        textarea.select();
        let successful = document.execCommand('copy');
        if (successful) {
          alert('Link copied to clipboard!');
        } else {
          throw new Error('Copy command was unsuccessful');
        }
        document.body.removeChild(textarea);
      } catch (err) {
```

```
            console.error('Fallback: Oops, unable to copy', err);
        }
    }
};

const navigateTo = (url) => {
    navigate(url);
};

return (
    <div className='base-unfixed-footer'>
    <div className='base-unfixed-footer-content'>
    <div className='my-survey-page'>
        <TopPanelBase/>
        <div className='my-survey-top'>
        <h1>My EmPolls</h1>
        <p>{userInfo ? userInfo.name: "User"}, you can see your Polls below:</p>
            <button
                onClick={() => setActiveTab('created')}
                style={{
                    backgroundColor: activeTab === 'created' ? '#720a7d': '#b115b7',
                    color: 'white',
                    padding: '10px',
                    margin: '5px',
                    border: 'none',
                    borderRadius: '5px'
                }}>
                My Created EmPolls
            </button>
            <button
                onClick={() => setActiveTab('taken')}
                style={{
                    backgroundColor: activeTab === 'taken' ? '#720a7d': '#b115b7',
                    color: 'white',
                    padding: '10px',
                    margin: '5px',
                    border: 'none',
                    borderRadius: '5px'
                }}>
                My Taken EmPolls
            </button>
        </div>
        <div className='my-survey-all'>
            {activeTab === 'created' && (
                <ul>
                    {createdSurveys.map((survey, index) => (
```

```
                    <li key={survey.survey_id}>
                        <div className='my-survey-all-unit'>
                            <p>"{survey.title}"</p>
                            <div className='my-survey-all-unit-buttons'>
                                <button  className='base-iconed-button'  onClick={()  =>
copyToClipboard(`${reactUrl}/survey/${survey.hash}`)}>
                                    Copy<img  src={copy_icon}  alt='copy-survey-link'
className="icon"/>
                                </button>

                                <button  className='base-iconed-button'  onClick={()  =>
navigateTo(`/edit-survey/${survey.survey_id}`)}>
                                    Edit<img  src={edit_icon}  alt='edit-survey'
className="icon"/>
                                </button>

                                <button  className='base-iconed-button'  onClick={()  =>
navigateTo(`/survey-results-admin/${survey.survey_id}`)}>
                                Results<img  src={stats_icon}  alt='results-survey'
className="icon"/>
                                </button>

                            </div>
                        ))}
                </ul>
                )}
            </div>
                {activeTab === 'taken' && (
                    <ul>
                        {takenSurveys.map((taken, index) => (
                            <div className='my-survey-all-unit'>
                                <li key={taken.survey_id}>
                                    <p>"{taken.title}"</p>
                                    <div className='my-survey-all-unit-buttons'>
                                        <button  className='base-iconed-button'  onClick={()  =>
navigateTo(`/survey-results/${taken.survey_id}`)}>
                                            Results<img  src={stats_icon}  alt='my-results-survey'
className="icon"/>
                                        </button>
                                    </div>
                                </li>
                            </div>
                        ))}
                    </ul>
                )}
            </div>
```

```
        </div>
        </div>
        <BotPanel is_fixed={false}/>
        </div>
    );
}
export default MySurvey;

const SurveyResults = () => {
  const { id } = useParams();
  const [data, setData] = useState(null);
  const [similar, setSimilar] = useState(null);

  useEffect(() => {
    const fetchResults = async () => {
      try {
        const response = await axios.get(`${serverUrl}/survey/${id}/results`, {
          headers: {
            Authorization: `Bearer ${localStorage.getItem('authToken')}`,
          },
        });
        setData(response.data);
      } catch (error) {
        console.error('Error fetching survey results:', error);
      }
    };

    const fetchSimilar = async () => {
        try {
          const response = await axios.get(`${serverUrl}/survey/${id}/all-results`, {
            headers: {
              Authorization: `Bearer ${localStorage.getItem('authToken')}`,
            },
          });
          setSimilar(response.data);
        } catch (error) {
          console.error('Error fetching survey results:', error);
        }
      };
      fetchSimilar();
      fetchResults();
  }, [id]);

  if (!data || !similar) {
    return <div>Loading results...</div>;
  }
```

```
  return (
    <div className='base-unfixed-footer'>
    <div className='base-unfixed-footer-content'>
    <div className="result-survey-main">
      <TopPanelBase/>
      <div className='result-survey-top'>
        <h1>Poll Results</h1>
        <h2 style={{fontSize: '30px'}}>{data.survey.title}</h2>
        <p>{data.survey.description}</p>
      </div>
      <h3>Questions</h3>
      <div className='result-survey-ser'>
      {data.questions.map((question, index) => {
        let responseUser = [];
        question.responses.forEach((resp) => {
          responseUser.push(resp.value);
        });
        const responsesAll = similar[question.questionId];
        const colors = generateColors(question.emotions);
        return (
          <div key={question.questionId} className='result-survey-ser-unit'>
              <Question question={question}/>
              <p>Your response:</p>
              <ChartComponent                                responses={responseUser}
emotions={question.emotions} colors={colors}/>
              <p>Collective responses:</p>
              <ChartComponent                                 responses={responsesAll}
emotions={question.emotions} colors={colors}/>
          </div>
        );
      })}
      </div>
    </div>
    </div>
      <BotPanel is_fixed={false}/>
    </div>
  );
};
export default SurveyResults;

function SurveyPage() {
  const { surveyHash } = useParams();
  const [survey, setSurvey] = useState(null);
  const [emotions, setEmotions] = useState({});
  const [isLoading, setIsLoading] = useState(true);
```

```
const [error, setError] = useState(null);
const [submitted, setSubmitted] = useState(false);
const [isSubmitting, setIsSubmitting] = useState(false);
const navigate = useNavigate();

useEffect(() => {
  const fetchSurvey = async () => {
    try {
      const response = await axios.get(`${serverUrl}/survey/take/${surveyHash}`);
      if (response.status !== 200) {
        throw new Error('Survey not found');
      }
      const data = response.data;
      console.log(data);
      setSurvey(data);
      const initialEmotions = {};
      data.questions.forEach(question => {
        const questionEmotions = {};
        question.emotions.forEach(emotion => {
          questionEmotions[emotion] = 0;
        });
        initialEmotions[question.question_id] = questionEmotions;
      });
      setEmotions(initialEmotions);
      console.log(initialEmotions);
    } catch (err) {
      setError(err.message);
    } finally {
      setIsLoading(false);
    }
  };

  fetchSurvey();
}, [surveyHash]);

function handleEmotionChange(questionId, emotion, value) {
  setEmotions(prev => ({
    ...prev,
    [questionId]: {
      ...prev[questionId],
      [emotion]: value
    }
  }));
}

async function submitSurvey() {
```

```
    setIsSubmitting(true); // Set submitting state to true
    try {
      const authToken = localStorage.getItem('authToken');

      const response = await axios.post(
        `${serverUrl}/survey/submit`,
        {
          surveyHash,
          emotions
        },
        {
          headers: {
            Authorization: `Bearer ${authToken}`
          }
        }
      );

      if (response.data.success) {
        setSubmitted(true);
        navigate(`/survey-results/${survey.survey_id}`);
      }
    } catch (error) {
      console.error('Error submitting survey:', error);
    } finally {
      setIsSubmitting(false); // Set submitting state to false
    }
  }

  if (isLoading) {
    return <div>Loading...</div>;
  }

  if (error) {
    return <div>Error: {error}</div>;
  }

  if (!survey) {
    return <div>No survey data available.</div>;
  }

return (
  <div className='base-unfixed-footer'>
  <div className='base-unfixed-footer-content'>
  <div className='survey-page-main'>
    <div className='survey-page-top'>
      <h1>{survey.title}</h1>
```

```
      <p>{survey.description}</p>
    </div>
    <div className='survey-page-question-container'>
     {survey.questions.map((question) => (
      <Question
       key={question.question_id}
       question={question}
       emotions={emotions[question.question_id] || {}}
       handleEmotionChange={handleEmotionChange}
      />
     ))}
     </div>
     {!submitted && (
      <button onClick={submitSurvey} disabled={isSubmitting}>
       {isSubmitting ? 'Submitting...' : 'Submit Survey'}
      </button>
     )}
     {submitted && <div>Survey submitted successfully!</div>}
     <p></p>
    </div>
    </div>
    <TopPanelBase/>
     <BotPanel is_fixed={false}/>
    </div>
  );
}
export default SurveyPage;

export function TakeSurveyPage() {
  const [surveyHash, setSurveyId] = useState('');
  let navigate = useNavigate();

  const handleSubmit = (event) => {
   event.preventDefault();
   navigate(`/survey/${surveyHash}`);
  };

  return (
   <div className="take-survey-page">
    <TopPanelBase/>
    <h1 htmlFor="surveyId">Код опитування:</h1>
    <form onSubmit={handleSubmit}>
     <input
      id="surveyId"
      type="text"
      value={surveyHash}
```

```
          onChange={(e) => setSurveyId(e.target.value)}
          placeholder="Введіть унікальний ідентифікатор опитування"/>
        <button type="submit">Пройти</button>
      </form>
      <p></p>
      <BotPanel is_fixed={true}/>
    </div>
  );
}
export default TakeSurveyPage;
```

ДОДАТОК В

ГРАФІЧНА ЧАСТИНА

«WEB-ЗАСТОСУНОК ДЛЯ ПРОХОДЖЕННЯ ОПИТУВАНЬ. ЧАСТИНА 2. КЛІЄНТСЬКА ЧАСТИНА.»

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Ганевич В. М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Колодний В. В.
(прізвище та ініціали)
«07» 06 2024 р

Вінниця ВНТУ – 2024 рік

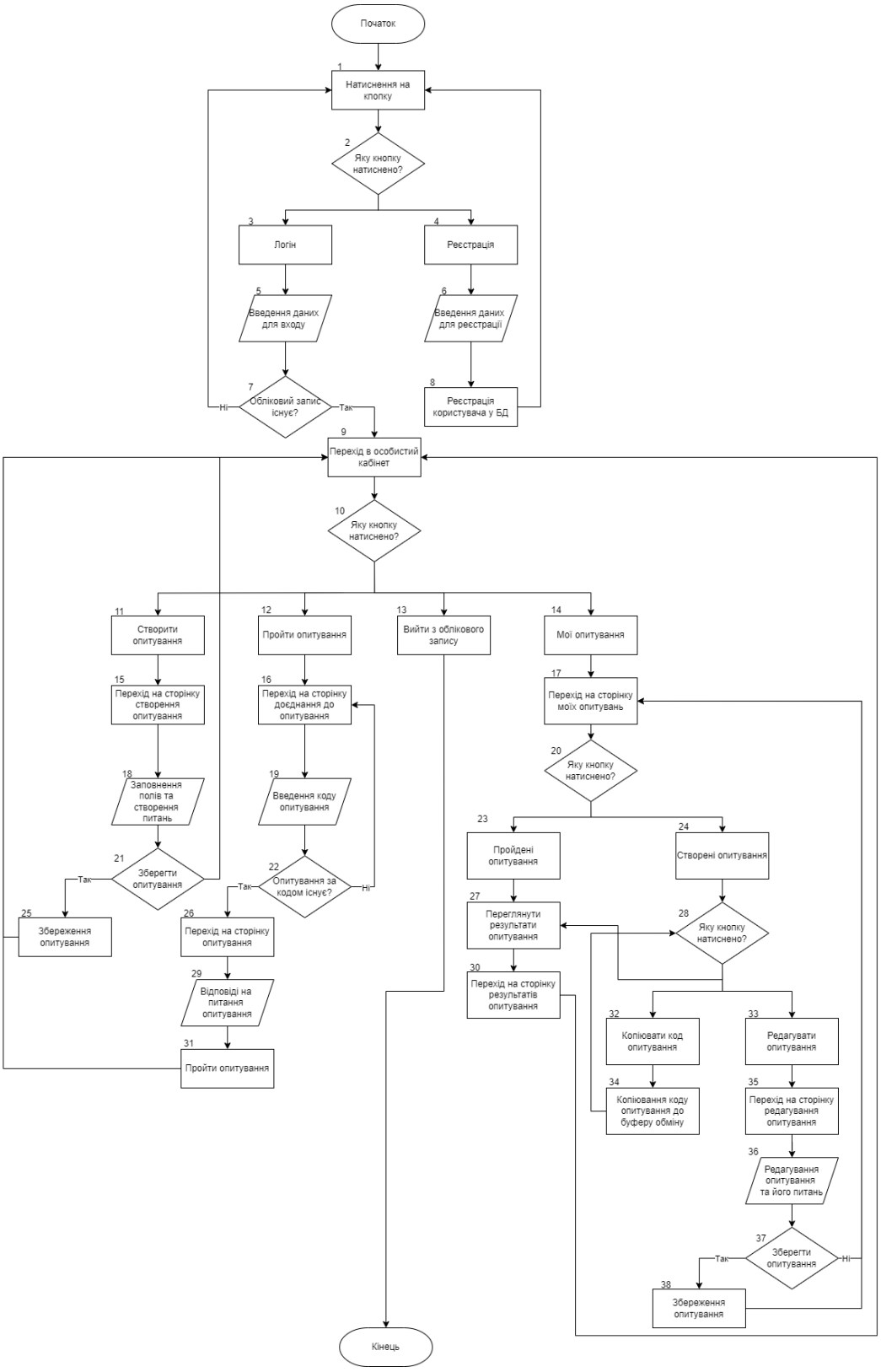

Рисунок В.1 – Схема алгоритму роботи WEB-застосунку для проходження опитувань

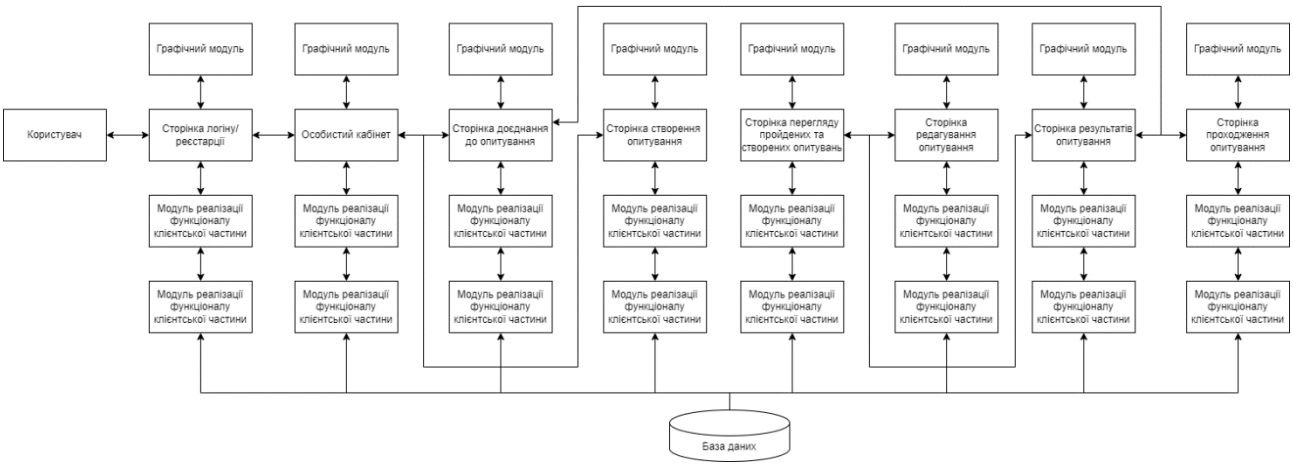

Рисунок В.2 − Загальна структура WEB-застосунку для проходження опитувань

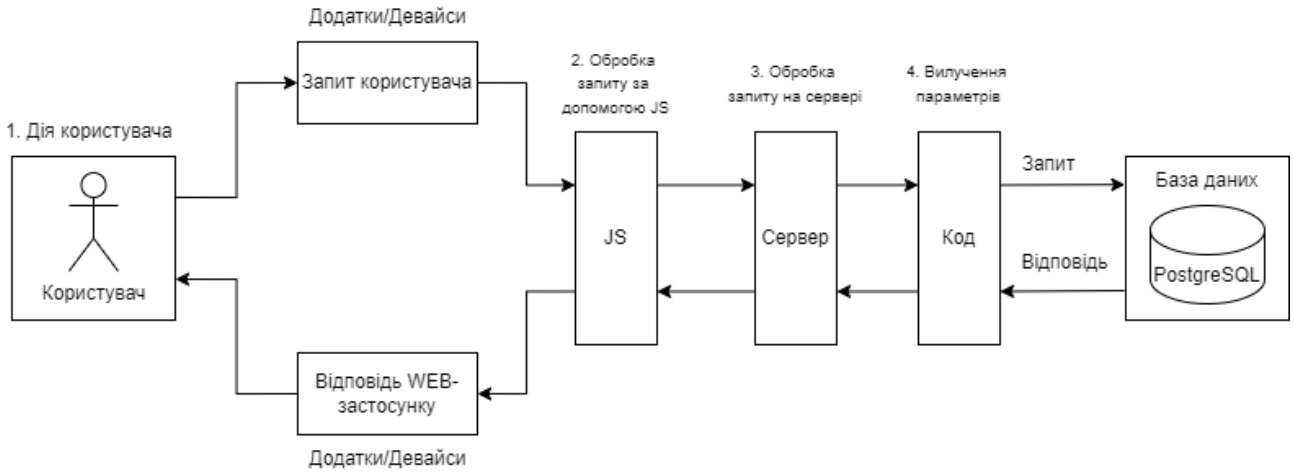

Рисунок В.3 – Структура принципу роботи WEB-застосунку для проходження опитувань

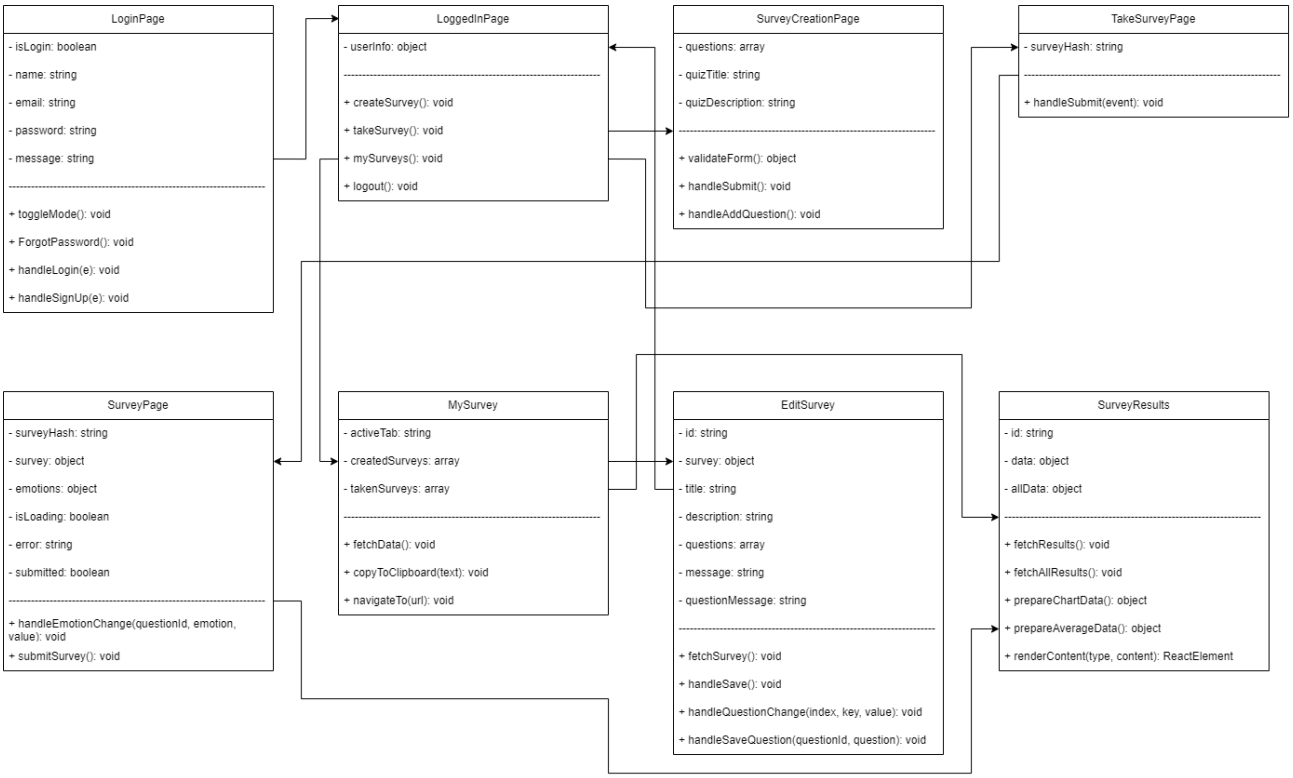

Рисунок В.4 – UML-діаграма класів WEB-застосунку для проходження опитувань

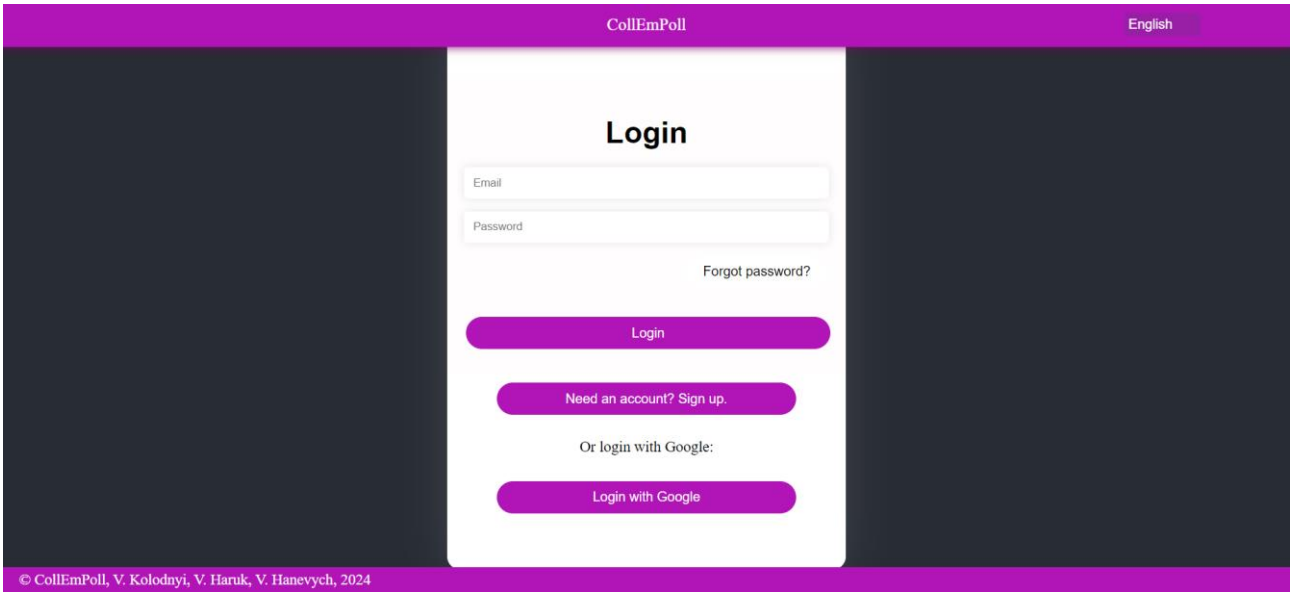

Рисунок В.5 – Сторінка входу в обліковий запис

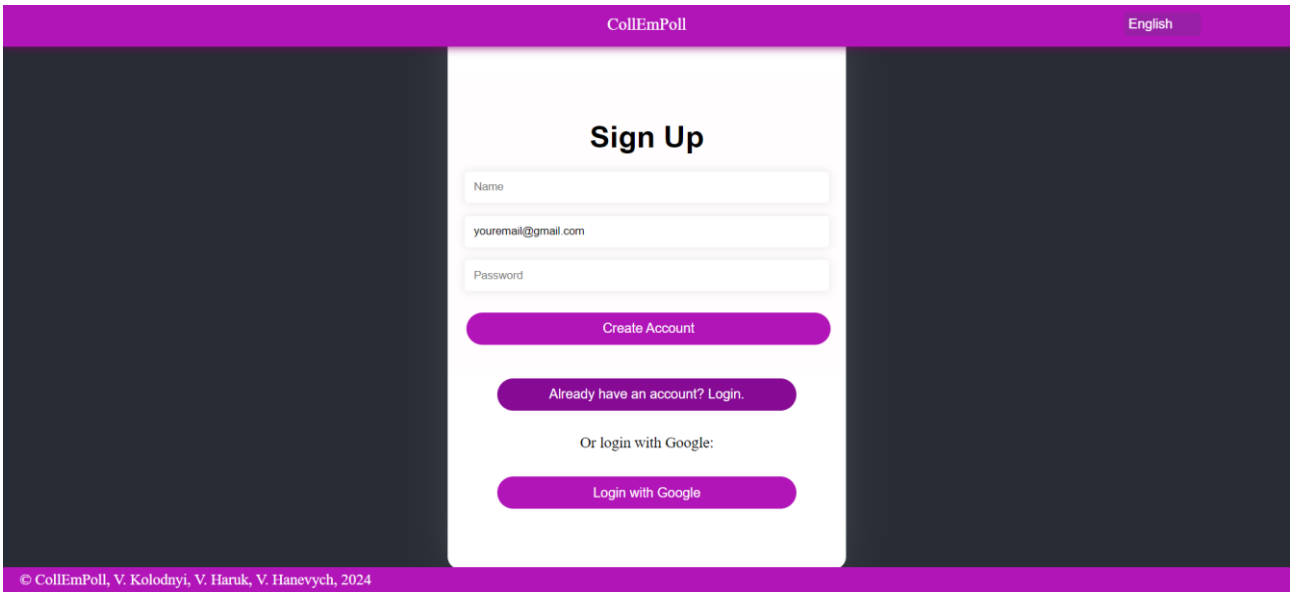

Рисунок В.6 – Сторінка створення облікового запису

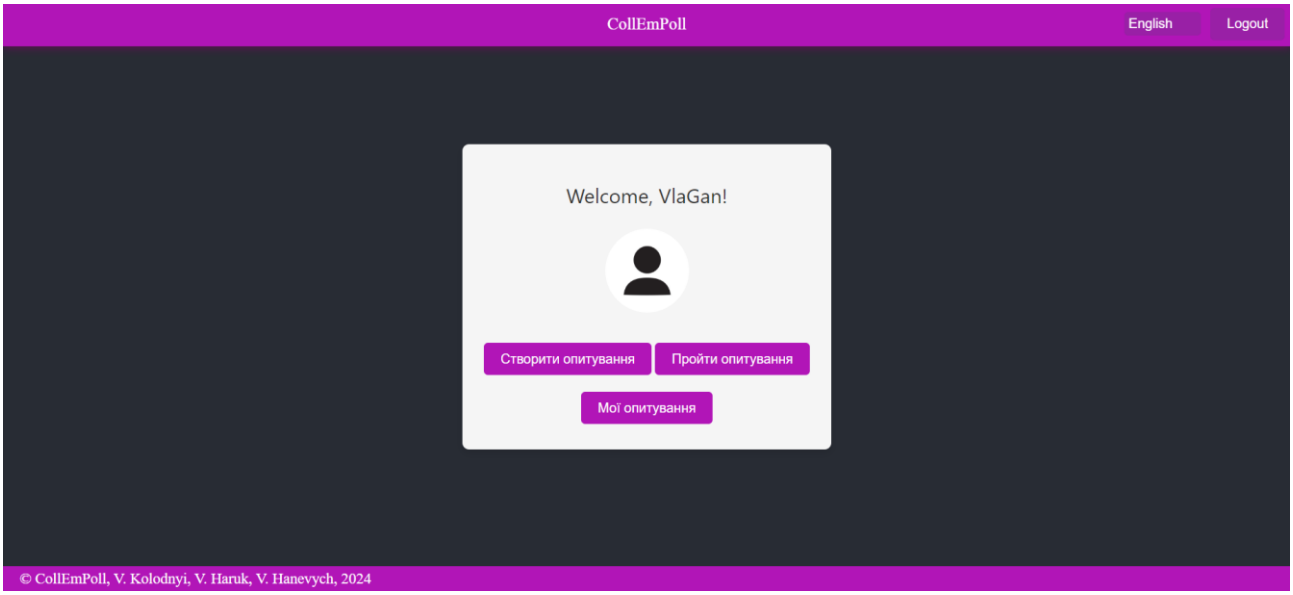

Рисунок В.7 – Особистий кабінет користувача

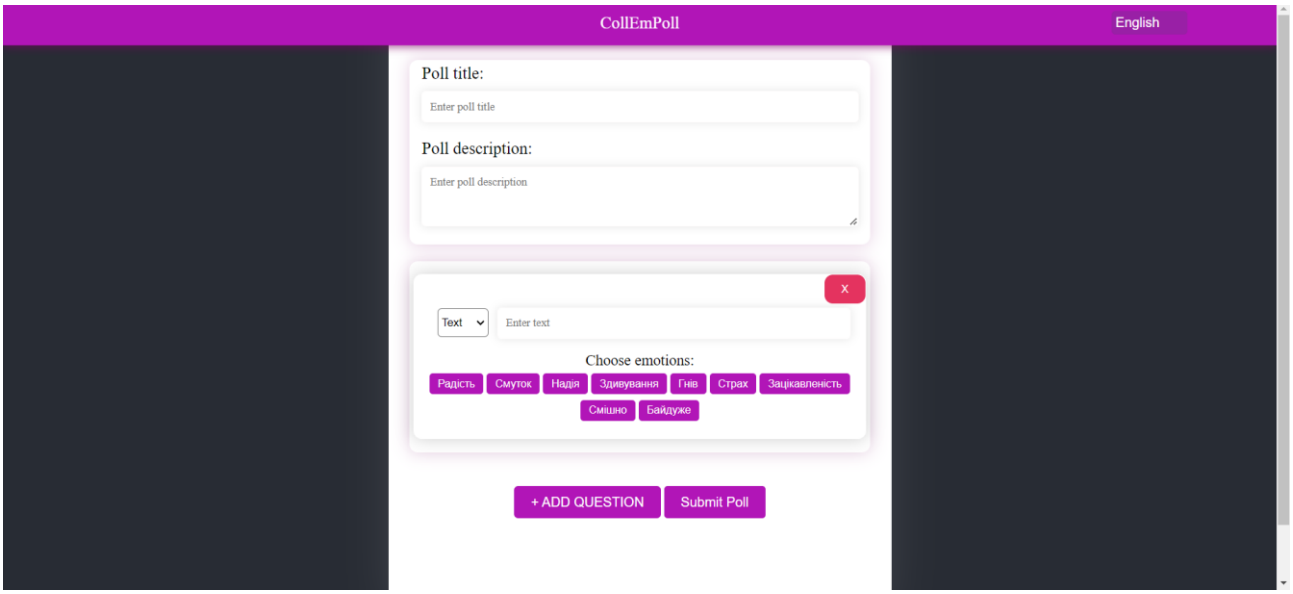

Рисунок В.8 – Сторінка створення опитування

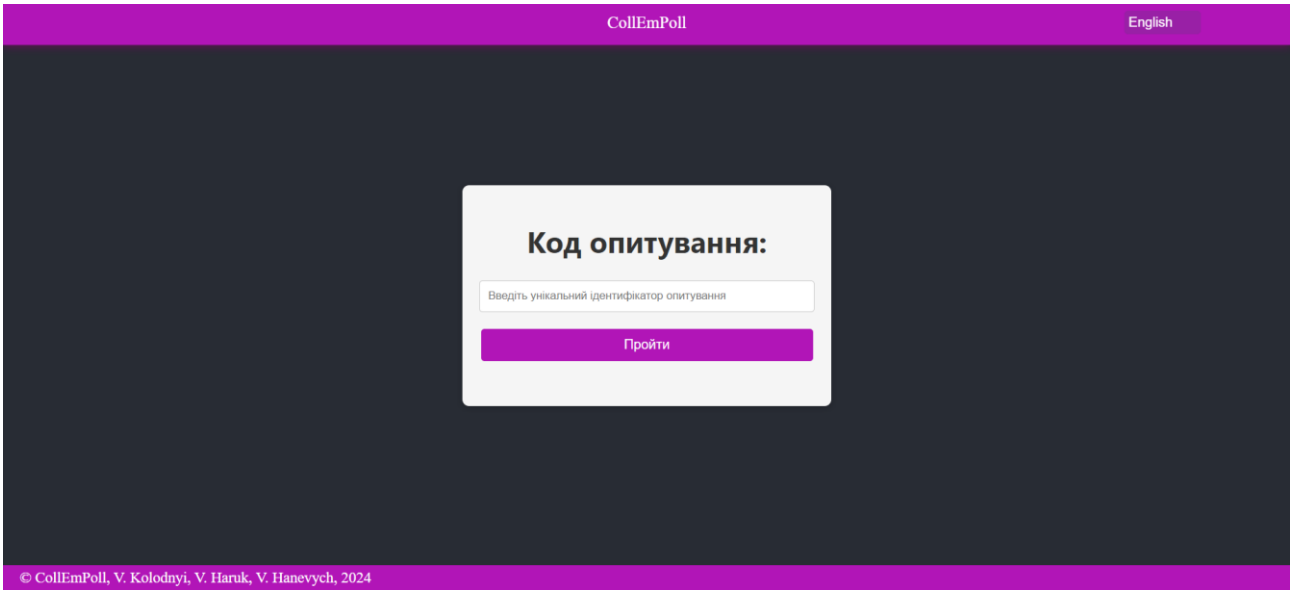

Рисунок В.9 – Сторінка доєднання до опитування

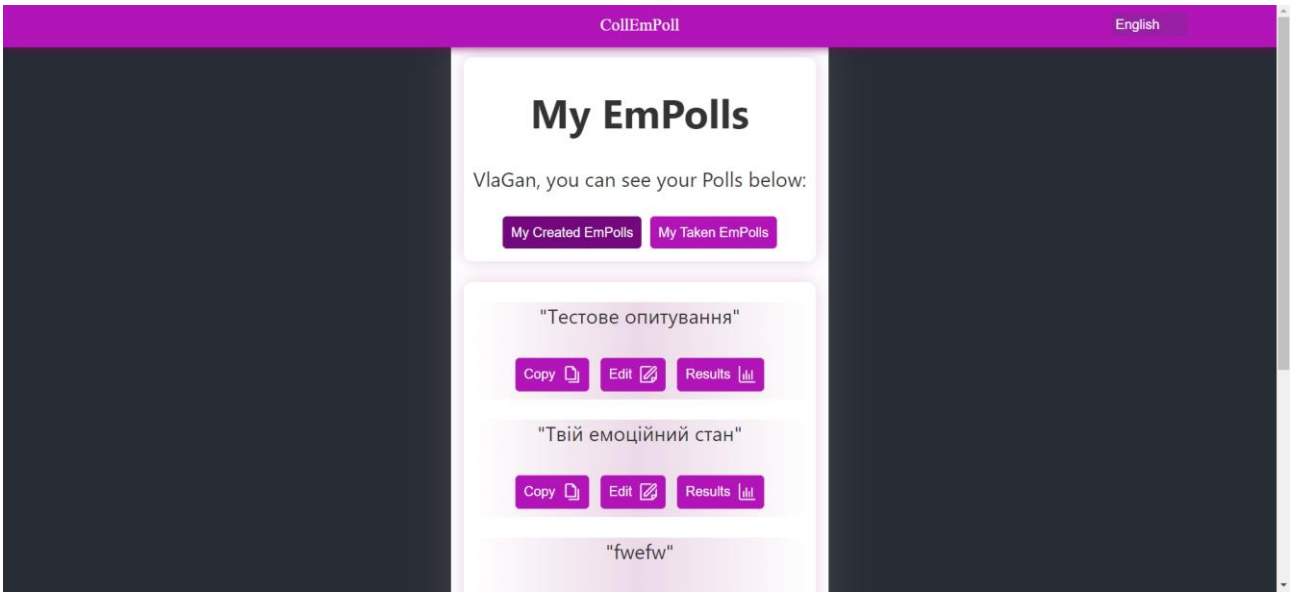

Рисунок В.10 – Сторінка створених та пройдених опитувань користувача

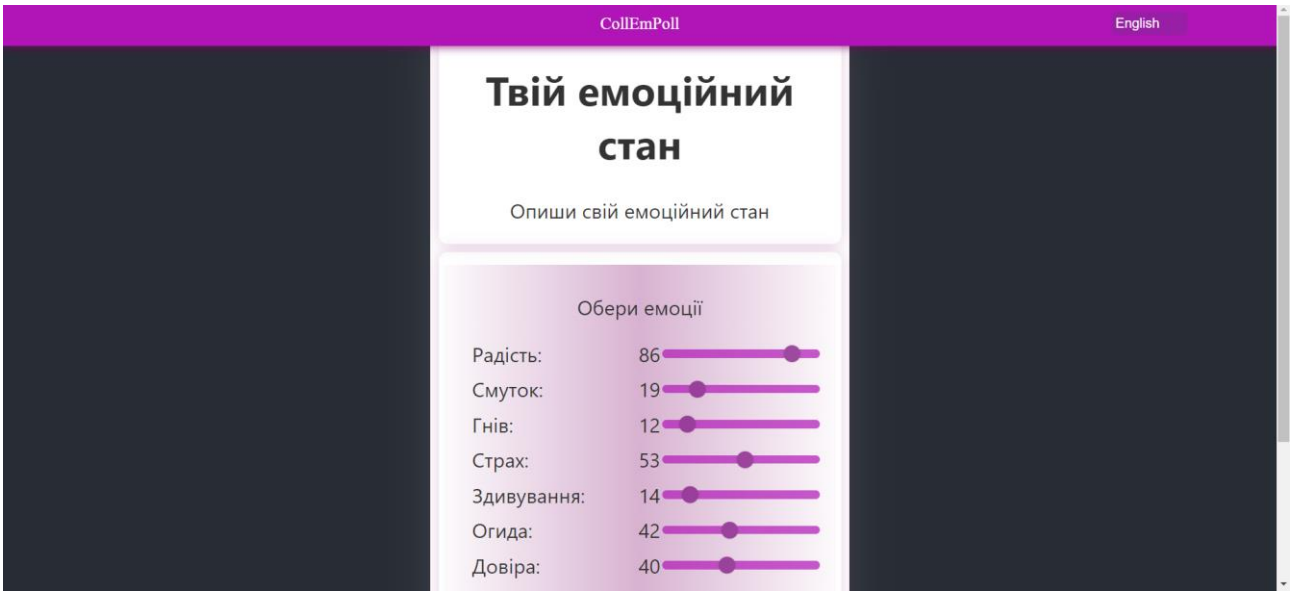

Рисунок В.11 – Сторінка проходження опитування

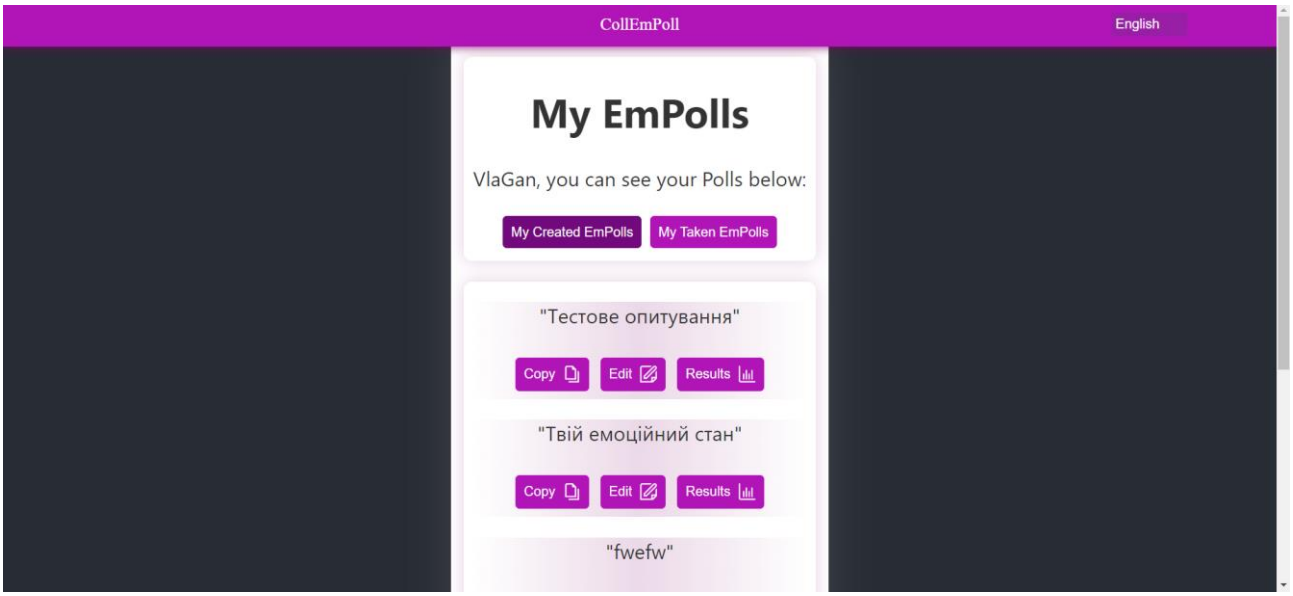

Рисунок В.12 – Пройдені користувачем опитування

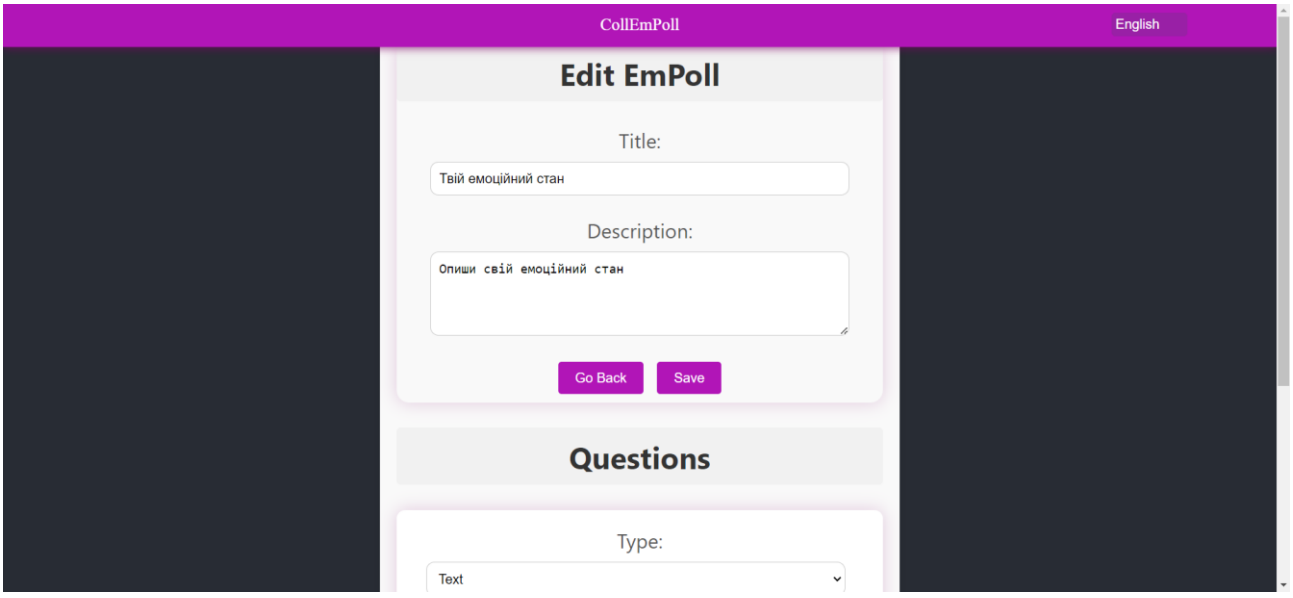

Рисунок В.13 – Редагування створених користувачем опитувань

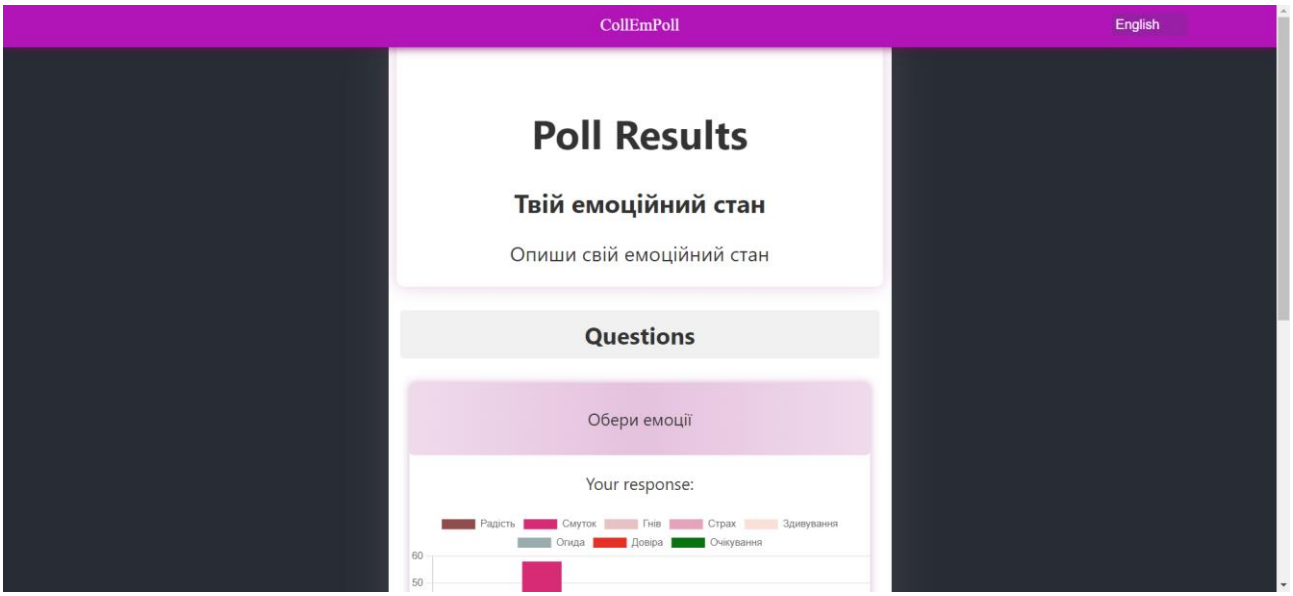

Рисунок В.14 – Результати опитування

Рисунок В.14 – Продовження

ДОДАТОК Г

Інструкція користувача

Для використання WEB-застосунку дял проходження опитувань, необхідно спочатку зареєструвати новий аккаут та увійти зі створеними даними, для чого потрібно на сторінці логіну (рис. Г.1) обрати реєстацію, створити новий аккаунт (рис Г.2), після чого повернутись на сторінку логіну, ввести дані та перейти в особистий кабінет (рис. Г.3).

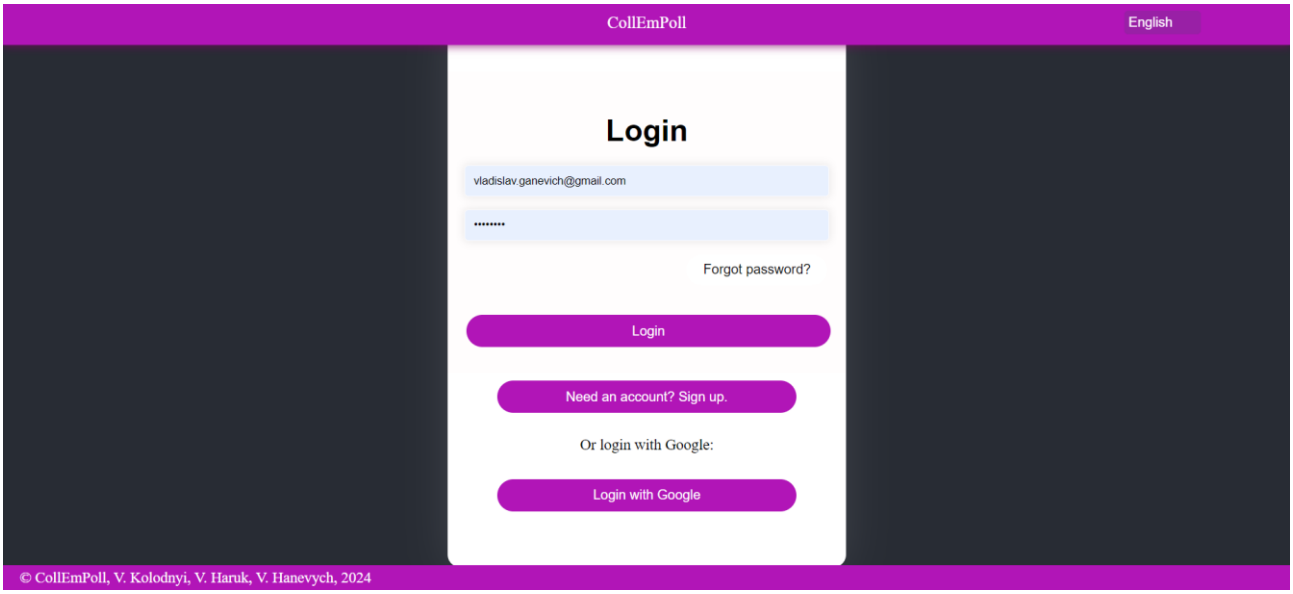

Рисунок Г.1 – Сторінка логіну

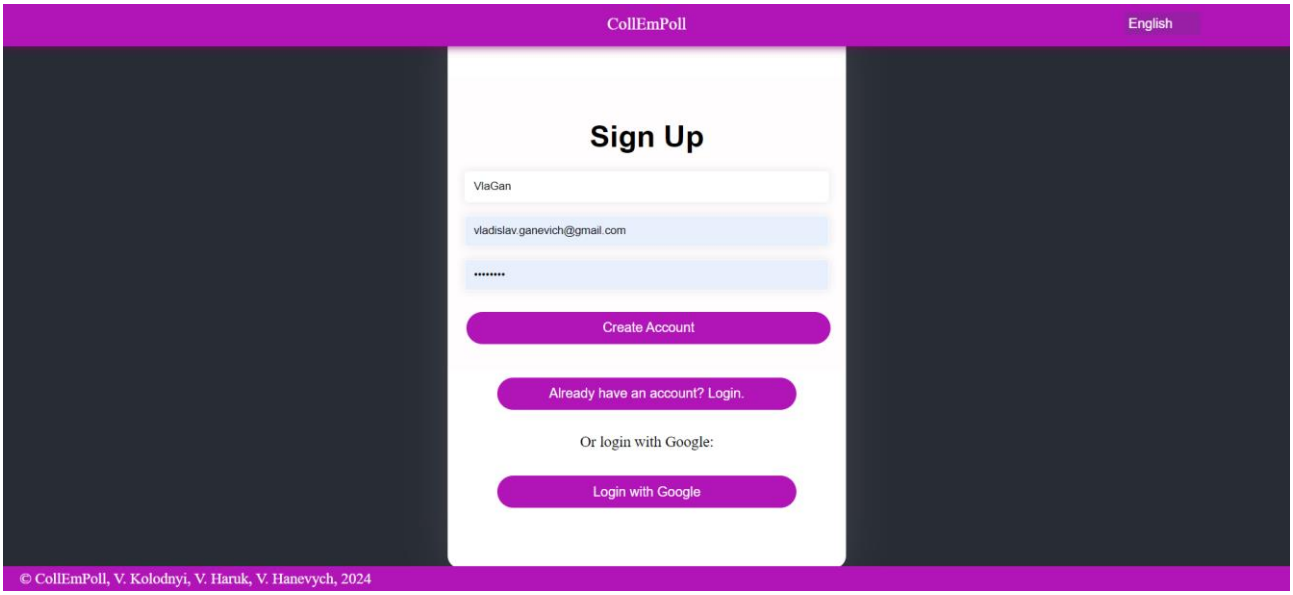

Рисунок Г.2 – Створення облікового запису

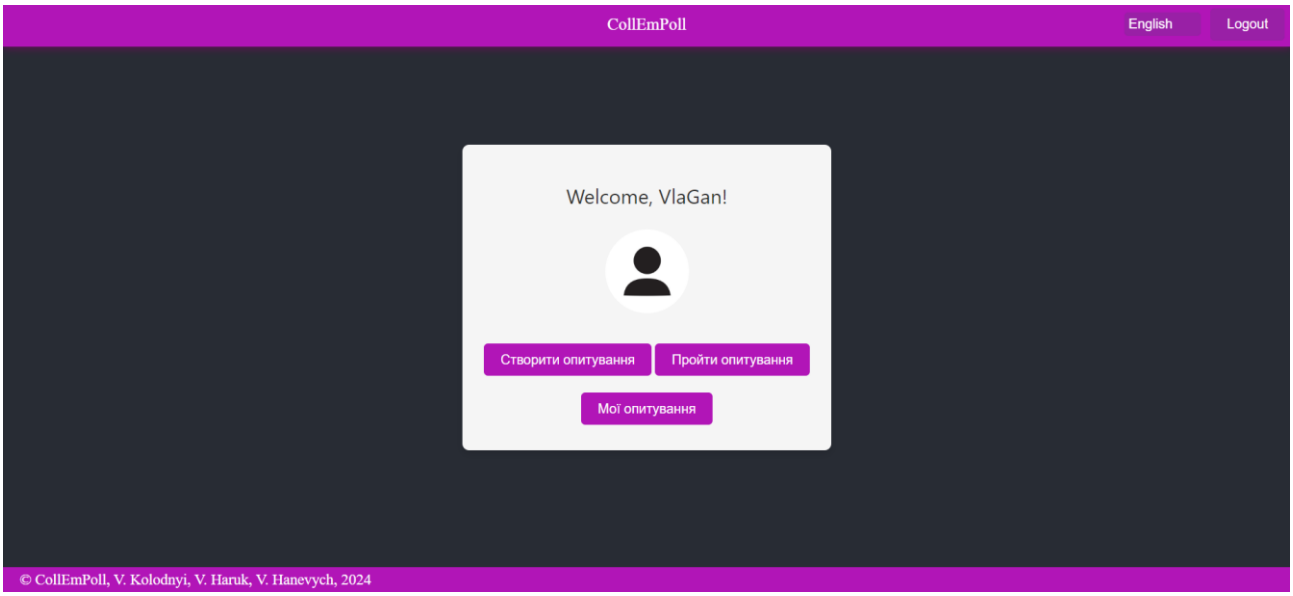

Рисунок Г.3 – Особистий кабінет користувача

З рисунку Г.3, можна помітити, що користувач має 3 основні можливості:

«Створити опитування» – дозволяє створити нове опитування, потрібне користувачеві. Якщо користувач обирає створення опитування, його перенаправить на сторінку створення опитування (рис. Г.4), де можна створити власне опитування, використовуючи своєрідний конструктор питань, після чого необхідно підтвердити створення опитування і користувача перенаправить на сторінку з інформацією про створене опитування (рис. Г.5);

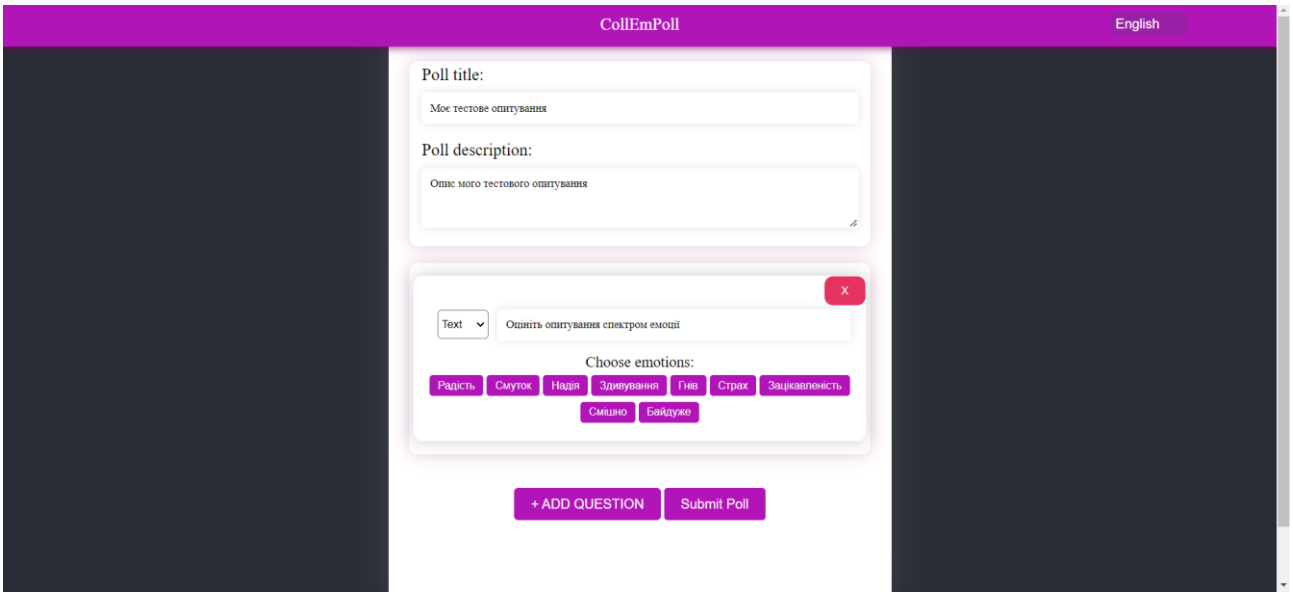

Рисунок Г.4 – Сторінка створення опитування

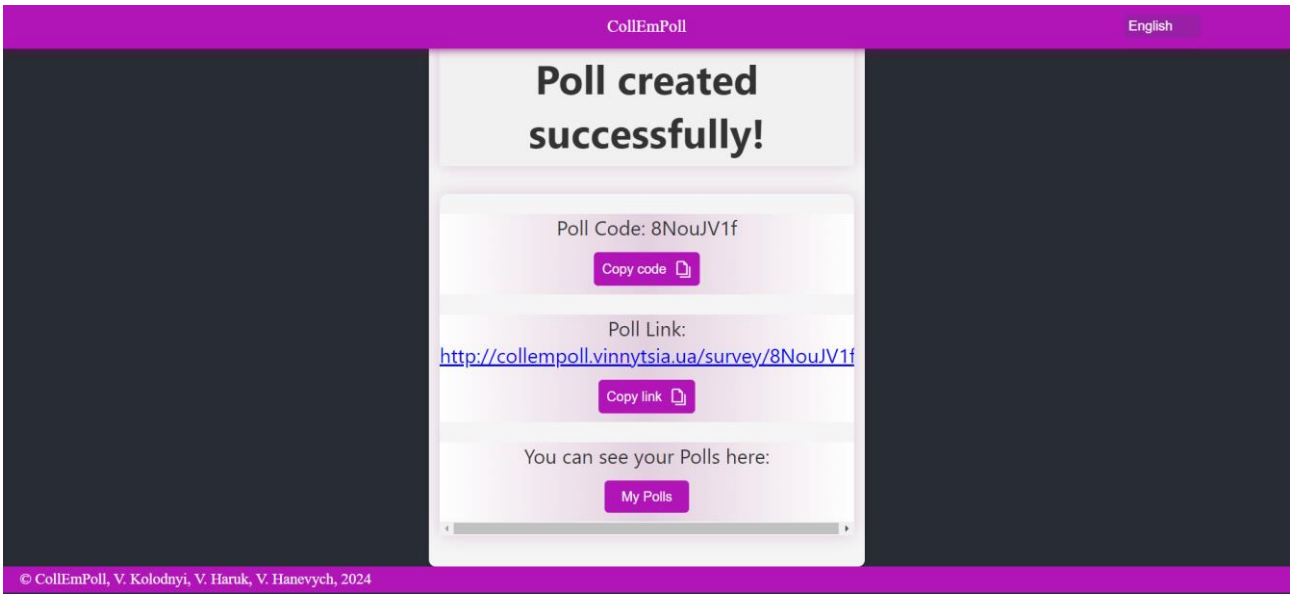

Рисунок Г.5 – Сторінка з інформацією про створене опитування

«Пройти опитування» – дозволяє доєднатись до опитування, маючи певний унікальний код опитування. При виборі даної опції користувача перенаправить на сторінку доєднання до опитування (рис. Г.6), на якій, ввівши унікальний код опитування і натиснувши кнопку підтвердження, можна пройти опитування (рис. Г.7);

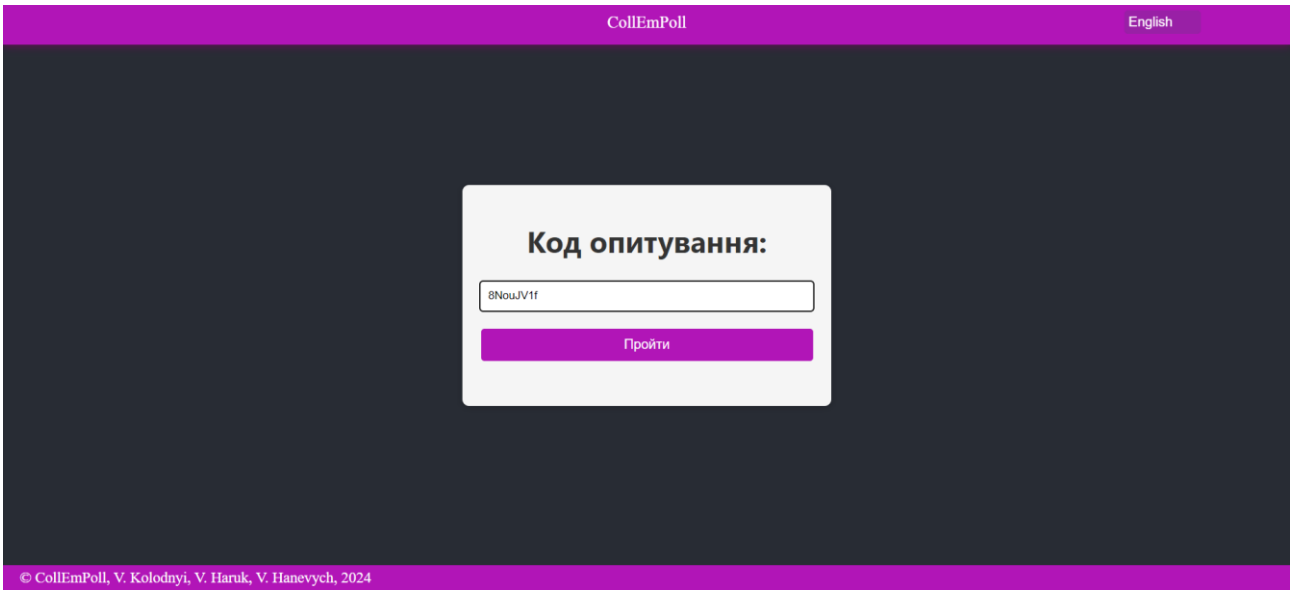

Рисунок Г.6 – Сторінка доєднання до опитування

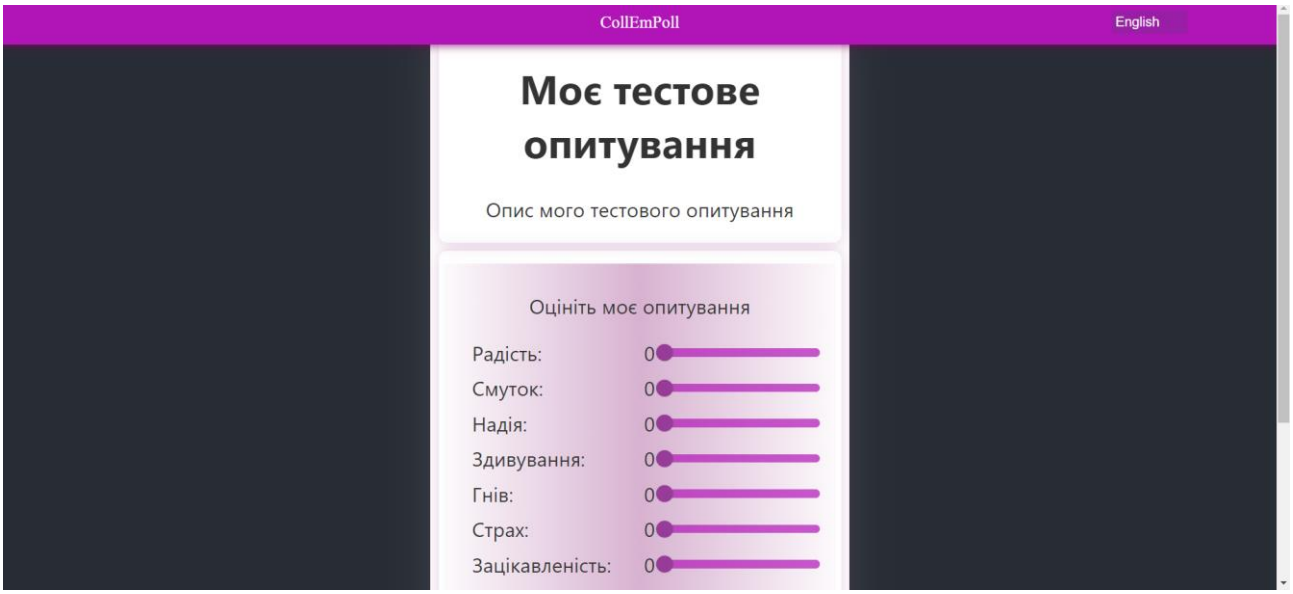

Рисунок Г.7 – Сторінка проходження опитування

«Мої опитування» – дозволяє переглянути створені та пройдені користувачем опитування. При виборі даної опції користувача перенаправить на сторінку створених та пройдених ним опитувань (рис. Г.8), на якій користувач зможе переглянути свої результати пройдених опитувань (рис. Г.9), а також редагувати створені опитування (рис. Г.10), скопіювати посилання на створене опитування і переглянути результати опитування для будь-якого користувача, який його пройшов (рис. Г.11).

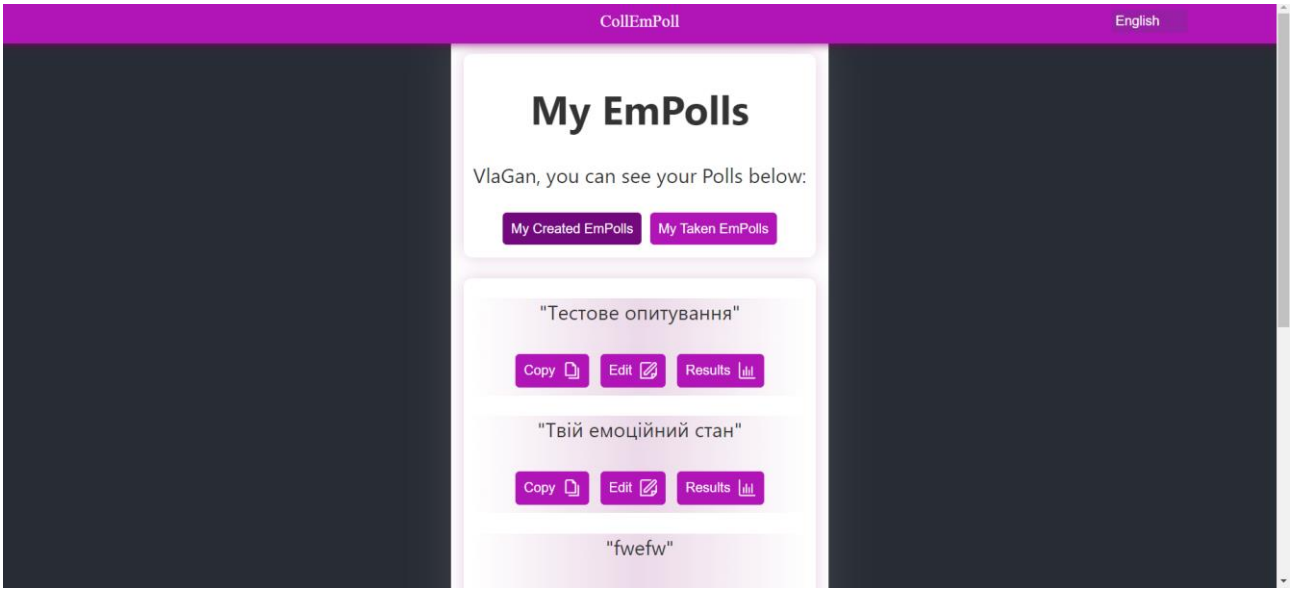

Рисунок Г.8 – Сторінка створених та пройдених користувачем опитувань

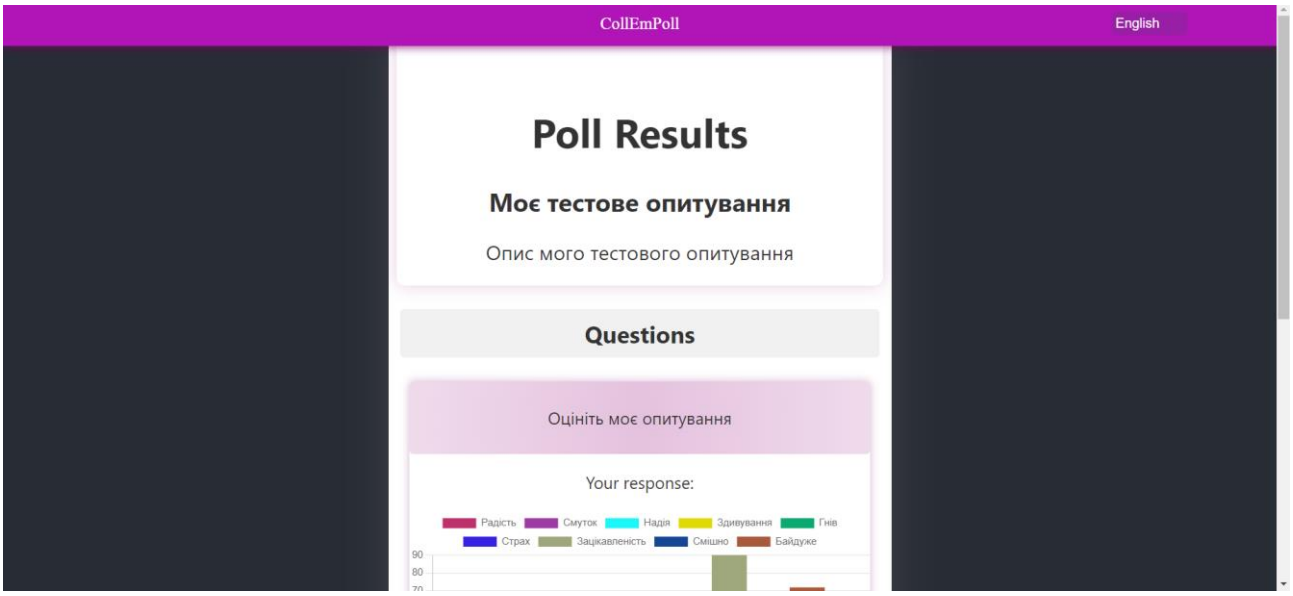

Рисунок Г.9 – Приклад результатів пройденого користувачем опитування

Рисунок Г.9 – Продовження

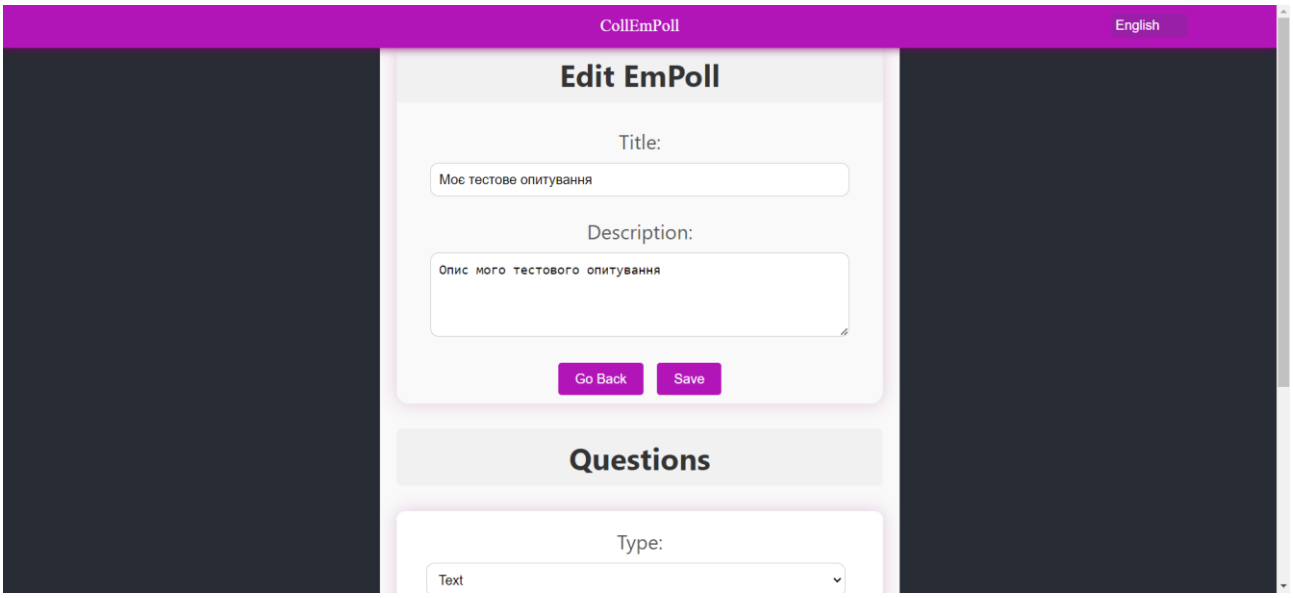

Рисунок Г.10 – Приклад редагування створеного користувачем опитування

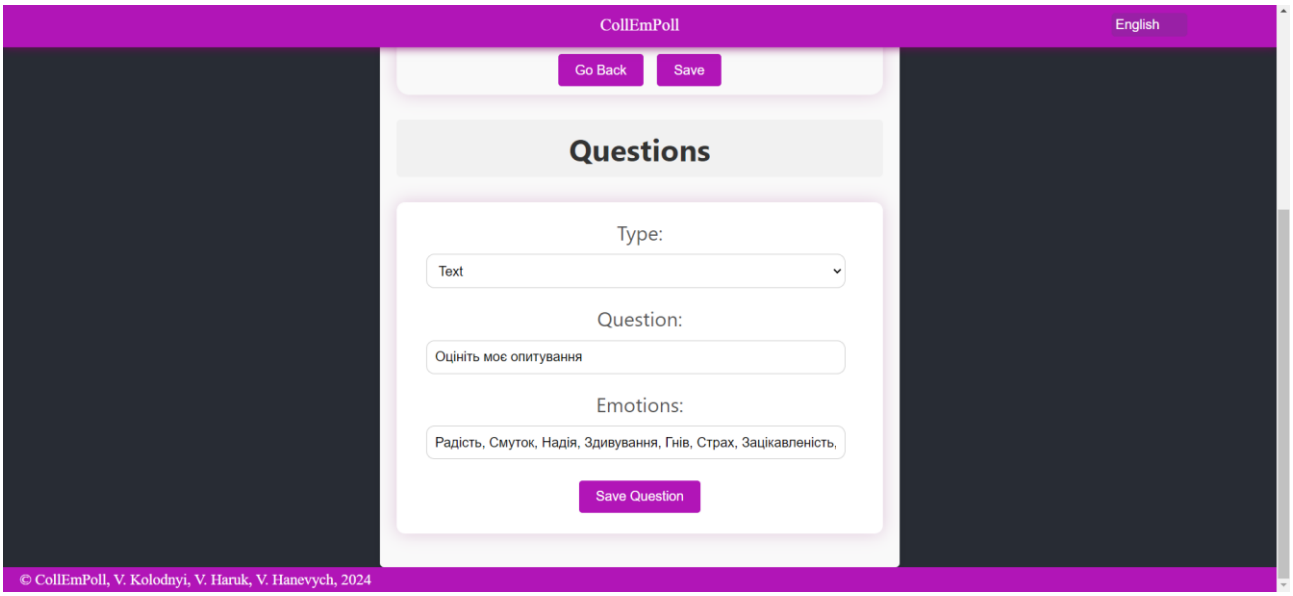

Рисунок Г.10 – Продовження

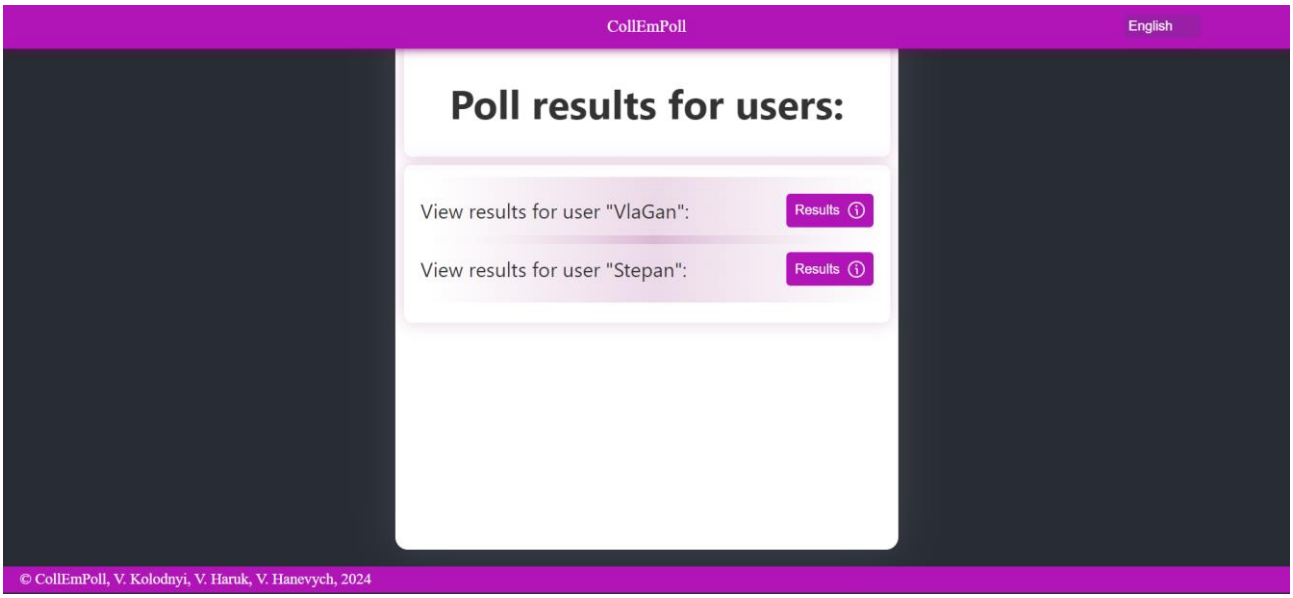

Рисунок Г.11 – Приклад перегляду результатів всіх користувачів, які пройшли опитування

З рисунку Г.11 можна помітити, що опитування пройшли декілька користувачів, для перегляду результатів кожного з них слід обрати кнопку результатів навпроти нікнейму користувача, після чого відкриється сторінка з розширеними результатами потрібного користувача в порівнянні з усіма опитаними (рис. Г.12).

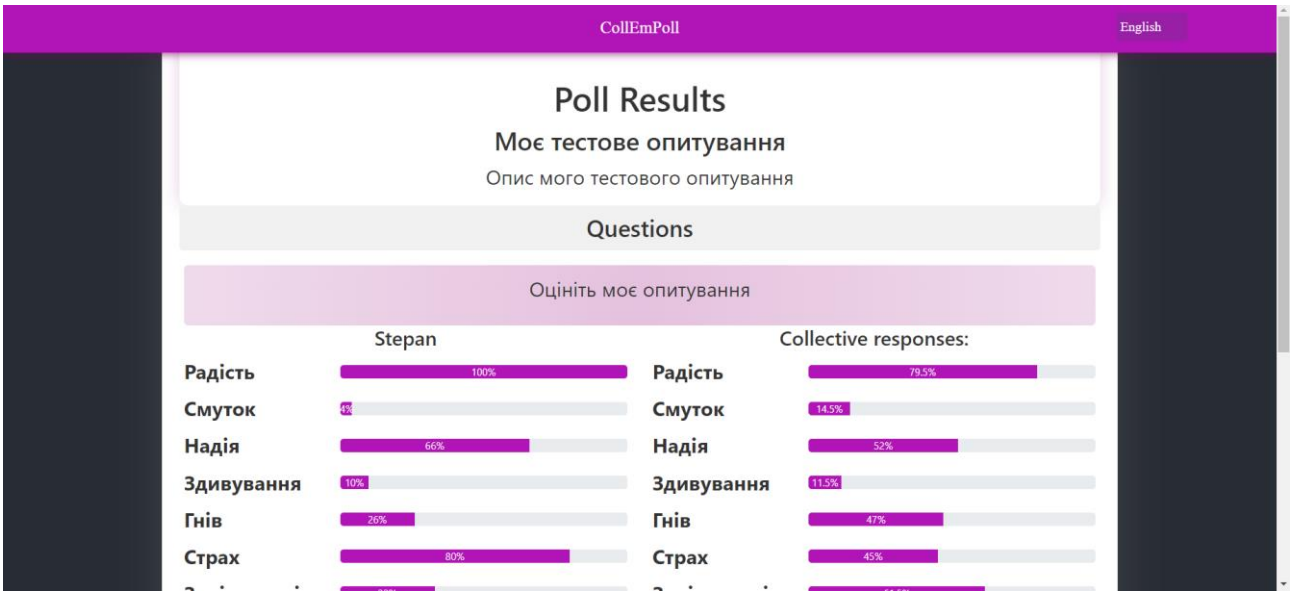

Рисунок Г.12 – Сторінка з розширеними результатами певного користувача в порівнянні з усіма опитаними

Рисунок Г.12 – Продовження