МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ВОЛОНТЕРСЬКИМИ РЕСУРСАМИ ТА ВЗАЄМОДІЇ З БЛАГОДІЙНИМИ ОРГАНІЗАЦІЯМИ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Віолетта ЛЕЖНЕНКО

«___»_____ 2025 р.

Керівник канд. фіз.-мат. наук, доцент _____ Інесса КУЛАКОВСЬКА

«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Лежненко Віолетти Сергіївни
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Вебзастосунок для управління волонтерськими ресурсами та взаємодії з благодійними організаціями

Керівник роботи: Кулаковська Інесса Василівна, доцент каф. ІІС, канд. фіз.-мат. наук, доцент.
(прізвище, ім'я, по батькові, посада, науковий ступінь, вчене звання)

Затверджена наказом ЧНУ ім. Петра Могили від « 18 » листопада 2024 р. №321

2. Строк представлення кваліфікаційної роботи «__» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні:
Вхідні дані: база даних для управління волонтерськими ресурсами, вимоги користувачів щодо функціоналу та інтерфейсу вебсистеми.
Очікуваний результат: розроблена вебсистема для управління волонтерськими ресурсами та взаємодії з благодійними організаціями, яка включатиме систему управління завданнями, моніторинг використання ресурсів на благодійність та систему комунікації.

4. Перелік питань, що підлягають розробці:

огляд та аналіз предметної сфери; постановка задачі; визначення методів для вирішення завдання; аналіз та вибір технологій розробки системи; розробка дизайн документу; налаштування БД та програмна реалізація вебзастосунку для управління волонтерськими ресурсами та взаємодії з благодійними організаціями; розробка алгоритму шифрування AES-256-GCM для безпечного зберігання важливих даних про користувача; інтеграція з платіжною системою Fondy для забезпечення можливості здійснення грошових переказів на благодійність; реалізація внутрішнього чату платформи для забезпечення комунікації; тестування основного функціоналу системи; визначення шляхів подальшого розвитку проєкту.

5. Перелік графічних матеріалів: рисунки, таблиці, формули, презентація

Керівник роботи _____ Інесса КУЛАКОВСЬКА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Віолетта ЛЕЖНЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «18» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Вебзастосунок для управління волонтерськими ресурсами та взаємодії з благодійними організаціями

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	виконано
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних систем, щодо управління волонтерськими ресурсами та взаємодії з благодійними організаціями	31.01.2025	01.03.2025	виконано
4	Огляд існуючих методів та технологій для вирішення поставленої задачі	02.03.2025	01.04.2025	виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	02.04.2025	15.05.2025	виконано
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	виконано
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	виконано
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	виконано
9	Доробка та остаточне оформлення КР	31.05.2025	04.06.2025	виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	11.06.2025	виконано

Керівник роботи _____ Інесса КУЛАКОВСЬКА

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Віолетта ЛЕЖНЕНКО

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану

«29» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 401 ЧНУ ім. Петра Могили

Лежненко Віолетти Сергіївни

на тему: «**ВЕБЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ВОЛОНТЕРСЬКИМИ РЕСУРСАМИ ТА ВЗАЄМОДІЇ З БЛАГОДІЙНИМИ ОРГАНІЗАЦІЯМИ»**

Актуальність дослідження підтверджується наявністю великої кількості волонтерів, відсутності цілісної системи для керування та комунікації благодійністю, адже більшість з аналогів спрямовано на грошову допомого, упускаючи з виду наявність великої кількості людей, що можуть допомагати фізично або матеріалами. Крім того, сьогоденна Україна наразі живе у часи війни, і волонтерська допомога рятує життя людей, допомагає з житлом, їжею та іншими побутовими потребами, тому покращення системи волонтерської діяльності та благодійності наразі є доволі важливим важелем для підтримки нації, що ще раз підтверджує, що тема має велике значення у розвитку країни.

Об’єктом роботи є система управління волонтерськими ресурсами та взаємодії з благодійними організаціями, її архітектура, функціональні можливості, алгоритми обробки даних, способи інтеграції з іншими сервісами.

Предметом роботи є набір технологій та методів розробки вебсистеми, яка дозволяє ефективно керувати волонтерськими ресурсами, взаємодіяти з благодійними організаціями, автоматизувати процеси збору даних, забезпечити комунікацію.

Метою цієї роботи є покращення системи управління волонтерськими ресурсами та благодійними організаціями, шляхом розробки вебзастосунку, враховуючи аналіз існуючих проблем даної предметної сфери.

В результаті виконання кваліфікаційної роботи проаналізовано предметну область проєкту, що охоплює волонтерську діяльність та роботу благодійних організацій, проведено дослідження волонтерства та благодійництва сьогоденної України, визначено основні функціональні та нефункціональні вимоги, необхідні

для реалізації вебзастосунку, підібрано необхідні технології та визначено методи для вирішення поставленої задачі, створено дизайн документ вебзастосунку, визначено необхідні вхідні дані, створено базу даних та налаштовано необхідні зв’язки між таблицями, розроблено вебплатформу для управління волонтерськими ресурсами та взаємодії з благодійними організаціями «HeartUA», проведено тестування створеної системи та визначено шляхи подальшого розвитку вебзастосунку.

Дана робота складається з 5 розділів. У першому розділі проаналізовано стан розвитку волонтерської діяльності в Україні та ознайомлено з діяльністю різних благодійних організацій для визначення потрібної системи комунікації між ними. Другий розділ передбачає дослідження методів та технологій, потрібних для створення вебплатформи, що відповідає меті роботи, для вибору найкращих варіантів, що забезпечить створення професійного вебзастосунку. Третій розділ присвячений проєктуванню, моделюванню вебплатформи для управління волонтерськими ресурсами та взаємодії з благодійними організаціями, опису вхідних даних та структури системи. Четвертий розділ передбачає опис програмної реалізації проєкту кваліфікаційної роботи, тестування розробленої вебсистеми для виявлення та виправлення недоліків, у разі їх виникнення, з метою забезпечення безпечної та безперебійної роботи системи вебзастосунку. У п’ятому розділі розглянуто і проаналізовано результати створеної платформи, а також визначення шляхів подальшого покращення системи. Загальний обсяг роботи – 134 сторінок. Кваліфікаційна робота містить 7 таблиць, 72 рисунків, 1 формулу, 5 додатків та 36 джерел посилань.

Ключові слова: веброзробка, вебзастосунок, Vue, PHP, JS, HTML, CSS, волонтер, благодійність, благодійна організація, метод AES-256-GCM.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Lezhnenko Violetta

“WEB APPLICATION FOR MANAGING VOLUNTEER RESOURCES AND INTERACTION WITH CHARITABLE ORGANIZATIONS”

The relevance of this study is confirmed by the large number of volunteers and the lack of a unified system for managing and communicating charitable activities. Most existing analogs are focused on financial aid, overlooking the presence of a large number of people who can contribute physically or provide material support. Furthermore, modern Ukraine is currently living through a time of war, where volunteer assistance saves lives, provides housing, food, and other basic needs. Therefore, improving the system of volunteer activity and charity is a significant lever in supporting the nation, which once again confirms the importance of this topic for the country's development.

The object of the study is a system for managing volunteer resources and interacting with charitable organizations — its architecture, functional capabilities, data processing algorithms, and methods of integration with other services.

The subject of the study is a set of technologies and methods for developing a web system that enables efficient management of volunteer resources, interaction with charitable organizations, automation of data collection processes, and provision of communication.

The purpose of this study is to improve the system for managing volunteer resources and charitable organizations by developing a web application, taking into account the analysis of existing problems in the given subject area.

As a result of this qualification work, the subject area of the project, covering volunteer activities and the work of charitable organizations, has been analyzed. A study of volunteering and charity in modern Ukraine has been conducted. Key functional and non-functional requirements necessary for implementing the web application have been

identified. Appropriate technologies and methods for solving the defined problem have been selected. A design document for the web application has been created, required input data identified, a database designed, and the necessary relationships between tables configured. A web platform for managing volunteer resources and interacting with charitable organizations «HeartUA» has been developed. The created system has been tested, and directions for further development of the web application have been defined.

This work consists of five chapters. The first chapter analyzes the state of development of volunteer activity in Ukraine and reviews the activities of various charitable organizations to identify the necessary communication system between them. The second chapter explores the methods and technologies required to create a web platform that meets the goals of the study and selects the most suitable options to ensure the development of a professional web application. The third chapter is devoted to the design and modeling of the web platform for managing volunteer resources and interacting with charitable organizations, describing input data and system structure. The fourth chapter presents the software implementation of the qualification project, testing of the developed web system to identify and fix possible issues, aiming to ensure secure and uninterrupted operation of the web application. The fifth chapter analyzes the results of the created platform and outlines possible directions for further system improvement.

The total length of the work is 134 pages. The qualification work includes 7 tables, 72 figures, 1 formula, 5 appendices, and 36 reference sources.

Keywords: web development, web application, Vue, PHP, JS, HTML, CSS, volunteer, charity, charitable organization, AES-256-GCM method.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ДСТУ – державний стандарт України

ЕК – екзаменаційна комісія

ІІС – інтелектуальні інформаційні системи

КР – кваліфікаційна робота

СУБД – система управління базами даних

API – Application Programming Interface

AWS – Amazon Web Services

CLI – Command-Line Interface

CMS – Content Management System

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

IDE – Intergated Development Environment

SMS – Short Message Service

SQL – Structured Query Language

UI – User Interface

ВСТУП

Зростання агресії з боку російських військ у ході захоплення нашої держави підкреслює важливість вдосконалення системи волонтерської діяльності та благодійних організацій. Це необхідно для ефективного забезпечення ресурсами наших військових і цивільних, які постраждали від дій агресора, що незаперечно підтверджує актуальність даної теми.

Волонтери кожного дня стикаються з рядом труднощів, та попри все вони готові допомагати всім, хто залишився сам на сам з бідою. Не дивлячись на те, що існує багато вебсистем для благодійництва та волонтерства, вони не забезпечують достатню комунікацію, адже кожен з них спрямований саме на певну допомогу, найчастіше грошову. Але ж є громадяни, що не мають фінансової можливості допомагати, але можуть запропонувати фізичну допомогу або допомогти певними матеріалами, а раніше створені аналоги не охоплює таких волонтерів. Саме тому, наявні системи потребують удосконалення.

Метою КР є детальне дослідження предметної області, а саме волонтерської діяльності, роботи благодійних організацій, та проблеми, усунувши які можна покращити роботу волонтерів та, як наслідок, поліпшити не тільки становище окремих людей, що потребують допомоги, а й усього народу. Для досягнення цілі виконано певний перелік завдань, до яких входить:

- огляд та аналіз предметної сфери;
- постановка задачі; визначення методів для вирішення завдання;
- аналіз та вибір технологій розробки системи; розробка дизайн документу;
- налаштування БД;
- програмна реалізація вебзастосунку;
- розробка алгоритму шифрування AES-256-GCM для безпечного зберігання важливих даних про користувача;
- інтеграція з платіжною системою Fondy для забезпечення можливості здійснення грошових переказів на благодійність;

- реалізація внутрішнього чату платформи для забезпечення комунікації;
- тестування основного функціоналу системи;
- визначення шляхів подальшого розвитку проєкту.

Розробка «Вебзастосунок для управління волонтерськими ресурсами та взаємодії з благодійними організаціями» була представлена на секції «Технічні науки», підсекції "Інтелектуальні інформаційні системи" на XXI Міжнародній науковій конференції «Ольвійський форум – 2025: стратегії країн Причорноморського регіону в геополітичному просторі», яка відбулася 16-21 червня 2025 року на базі Чорноморського національного університету імені Петра Могили у м. Миколаїв.

1 АНАЛІЗ НАПРЯМУ БЛАГОДІЙНИЦТВА ТА ВОЛОНТЕРСТВА В УКРАЇНІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної сфери

Предметна область даного проекту охоплює волонтерську роботу та роботу благодійних організацій.

Волонтер – людина, що на безоплатній основі виконує певні дії та вчинки, спрямовані на покращення життя громади та суспільства в цілому, допомагає тим, хто цього потребує та не просить винагороди за свою роботу [1]. Волонтер може стикнутися з рядом труднощів, що можуть унеможливлювати виконання ним належної допомоги. До них варто віднести наступні:

- корупція;
- недостача ресурсів;
- недостача фінансів;
- недостатньо людей;
- високе емоційне та фізичне навантаження;
- нерозуміння з боку суспільства;
- проблеми з логістикою;
- організаційні труднощі;
- небезпека волонтерської роботи.

Благодійна організація – суб’єкт права, що має на меті допомагати громадянам, що потребують допомоги, причому не очікують плату за свою роботу. Така організація має свої певні статутні документи, що чітко і ясно визначають одну або декілька сфер її діяльності, наприклад захист здоров’я, освіта, права людини, тощо), та згідно цього, описують її мету [2].

Якщо взяти до уваги той факт, що благодійність можна розділити на декілька основних видів, таких як надання грошової допомоги, надання певних послуг та громадські роботи, то можна в повною впевненістю сказати, що й благодійні

організації мають відрізнятися між собою за цим критерієм [2]. Отже, опираючись на Закон України «Про благодійну діяльність та благодійні організації», варто зазначити, що благодійну організацію можна юридично заснувати як благодійний фонд, благодійне товариство та благодійну установу [3].

Благодійний фонд – благодійна організація, що може бути створена одним або декількома засновниками, має учасників, які нею ж і керують, і які не зобов'язані передавати їй активи для досягнення благодійної мети. Така організація діє згідно статуту [3-4].

Благодійне товариство – благодійна організація, що заснована не менше ніж 2 особами та діє згідно статуту [3-4].

Благодійна установа – благодійна організація, що діє згідно установчого акту, має один або декілька засновників, що у свою чергу передають право користування активами для досягнення цілей організації, при цьому не беручи участь в управлінні [3-4].

Волонтерство, у сьогоденній Україні, посіло доволі важливе місце у житті багатьох громадян, адже воно допомагає не тільки підтримувати національний дух і надавати допомогу людям у критичній ситуації, а й допомагає здобути емоційну рівновагу, завести нові знайомства, отримати нові знання і можливості реалізувати себе у різних сферах життя суспільства, наприклад у організації свят, ярмарок, роботи з тваринами, рукоділля, тощо.

Для поточної статистики розвитку волонтерської діяльності було проведено опитування, що складалося з 11 питань на тему «Волонтерство та благодійність», використовуючи безкоштовне програмне забезпечення Google Forms [5]. В опитуванні взяло участь 60 громадян різного віку (76.7% з них особи 18-35 років, 11.7% – громадяни до 18 років, 8.3% – особи 35-70 років та 3.3% – громадяни, яким більше 70 років). Більша половина опитуваних – жінки (58.3%).

На питання про частоту участі у благодійних ярмарках, лише 6.7% опитуваних часто беруть у них участь, 25% опитуваних рідко відвідують такі заходи, 23.3% одноразово відвідувати благодійну ярмарку, 23.3% не брали участь

у подібному, але хотіли б, та 21.7% не відвідували і не мають бажання брати участі у подібних заходах. Отже більшість людей брали участь або хотіли б брати участь у благодійних ярмарках, що показує зацікавленість людей цією тематикою.

Серед опитуваних лише невелика кількість займається волонтерством (16.7%), але більша частина (48.3%) дали відповідь, що вони не є волонтерами, але хотіли б ними бути. Інші ж дали негативну відповідь, тобто 35% опитуваних і не хочуть займатися волонтерством. Більшість кажуть, що мають недостатньо часу займатися волонтерством, мають фінансові труднощі або просто не знають де зайти можливості.

Крім того, в опитувані піднімалася тема грошових внесків, що дало доволі очікувані результати, по яким більша частина опитуваних часто або рідно, але беруть участь і роблять фінансові пожертви для закриття певних зборів (26.7 – часто, 46.7 – рідко), 16.7% не мають фінансової можливості, але залюбки б допомагали, якби була така можливість, всі інші (10%) не робили грошові внески на благодійність і не хочуть. При цьому, найпопулярнішою тематикою зборів, у яких брали участь опитувані є військова допомога та допомога тваринам (притулкам), інші тематики, такі як допомога хворим, дітям, переселенцям набрали малий відсоток відповідей.

Довіра людей, що до діяльності благодійних організацій доволі відрізняються. Хоча більша кількість (56.7%) довіряють подібним установам, але все ж таки 30% категорично не мають до них довіри, і лише 13.3% або довіряють лише деяким з них, або просто намагаються проаналізувати діяльність тої чи іншої організації перед тим як, наприклад, надсилати їм кошти на щось.

Більшість опитуваних (50%) вважає, що якщо б вони і займалися волонтерством, то при якійсь організації, 35% хотіли б бути самостійними волонтерами, а всі інші не бажають займатися подібною діяльністю.

Що до онлайн платформ для благодійності думки опитуваних розділилися, 45% довіряють подібним веб сторінкам, а 41% категорично ні, інші ж опитувані відповіли, що в цілому довіряють лише перевіреним.

Додатково до цього розглянуто питання проблем, з яким стикаються волонтери у своїй діяльності. Не дивлячись на те, що серед опитуваних було не так багато волонтерів, але всі рівно їх думка важлива для забезпечення цілісності системи та, можливо, у подальшому допоможе залучити більше громадян до благодійної діяльності. В принципі, усі з представлених проблем набрало багато голосів опитуваних, але серед них варто виділити наступні, скажімо найпопулярніші, а саме: брак фінансування, недостача ресурсів та корупція. Меншу популярність отримали проблеми більш психологічного характеру, тобто безпека волонтерів, високе психологічне та фізичне навантаження, організаційні труднощі, тощо.

На питання «Які функції благодійної платформі були б корисними?» більшість голосів набрала відповідь про прозорість фінансів, друге і третє місце у рейтингу відповідей займає функція легкого пошуку волонтерських можливості та зручні варіанти грошових переказів на благодійність. Це питання має доволі важливе значення, адже безпосередньо допоможе під час складання технічного завдання проєкту.

І на останок опитувані поділилися, що б їх мотивувало стати волонтером. Для більшості мотивацією є вдячність людей, особиста історія та досвід, хоча доволі багато хто вказав на приклад знайомих та друзів. Найменш популярним, на диво, є відповідь про додаткові бонуси та нагороди.

Отже, проаналізувавши проведене опитування, можна сказати, що доволі багато людей, які мають бажання допомагати, хочуть в той чи іншій мірі бути волонтером, але стикаються з рядом проблем, тому статистику відповідей цього опитування буде враховано при програмній реалізації проєкту кваліфікаційної роботи.

1.2 Огляд та аналіз наявних аналогів та публікацій

Подібна тема вже була неодноразово висвітлена, зокрема студент КПІ ім. Ігоря Сікорського, Миргородський Н. В., який у своїй кваліфікаційній

роботі [6] поставив собі за мету створення веб сервісу збору та розподілу гуманітарної допомоги для покращення комунікації між волонтерами, та людьми. Він провів детальний аналіз проблем, пов’язаних з гуманітарною допомогою, виконав всі етапи розробки програмного продукту, використавши при цьому такі мови програмування, як Java, JavaScript, фреймворк React та середовище розробки IntelliJ Idea. В результаті створений програмний застосунок виконаний у рамках поставлених вимог та відповідно поставленої мети роботи, всі функціональні складові протестовані та демонструють безперебійну роботу.

Трішки з іншої сторони розглянув тематику волонтерства Сергієнко С. В., студент Державного торговельно-економічний університету, у своїй кваліфікаційній роботі [7]. Він поставив собі завдання створити веб-орієнтовний застосунок благодійної організації для покращення здійснення пожертвування на допомогу та залучення нових спонсорів. Використавши певні програмні засоби та технології, такі як мова програмування PHP, HTML, CSS, середовище розробки SublimeText, він зміг досягти поставленої мети, тобто сайт відповідає поставленим вимогам, хоча й має просту реалізацію, без додаткових функціональних ознак.

Розглянув цю тему з точки зору проблематики допомоги переселенцям у своїй кваліфікаційній роботі Пирогов В. І., студент Університету митної справи та фінансів, та взявся за створення вебзастосунку для допомоги волонтерам та ВПО «UAHelper» [8]. Він використав такі мови програмування, як C#, TypeScript, JavaScript, фреймворк Angular певні середовища розробки, до яких входять PhpStorm, Microsoft Studio Code, тощо. В результаті проведення дослідження та реалізації застосунку вирішення поставленої проблематики автор досяг поставленої мети, застосунок працює у повній мірі, виконуючи всі необхідні функції поставленої задачі.

Подібне бажання покращити систематизацію роботи волонтерів мала й студентка Маналі Сурeш Гейт, що присвятила цьому свою кваліфікаційну роботу ступеня магістра [9]. Проаналізувавши недоліки роботи волонтерів у рідному місті

Сакраменто, вона встановила, що робота волонтерських спільнот значно повільна через те, що вся координація волонтерів проводиться координаторами вручну, за допомогою телефону та електронної пошти, тому авторка вирішила посвятити свою роботу саме створення такої системи. Вона використала платформу Microsoft .NET Framework 3.5, середовище розробки Microsoft Visual Studio .NET 2008, мову програмування C#, СУБД для зберігання даних Microsoft SQL Server 2005 Express Edition, яка є безкоштовною, та Ez Texting Server API для надсилання повідомлень через БД. Як результат, вона створила цілісну систему координації волонтерів з можливістю реєстрації, авторизації волонтерів та координаторів, можливістю створення подій на керування ними. Хоча проєкт не має стилю, тобто спрямований безпосередньо на результат роботи обміну даними між користувачами, він повністю викує поставлені вимоги та відповідає меті.

Тематика збору коштів на благодійність доволі часто підіймається у сучасному світі, зокрема й у кваліфікаційній роботі Андруховича М. А., який вирішив створити вебзастосунок саме для збору пожертв [10]. Він обрав доволі незвичний шлях для вирішення поставленої цілі, використавши при цьому мову програмування Ruby, фреймворк Ruby on Rails та середовище розробки RubyMine. Розглянувши декілька аналогічних проєктів він виокремив головні позитивні моменти для свого проєкту, визначився з недоліками того чи іншого аналогу та, точно визначивши вимоги до вебсервісу, зміг створити цілісну систему для збору коштів на благодійність.

Крім аналогічних публікацій доречно було б проаналізувати й аналогічні вебсервіси. Одним з найбільших і найпопулярніших на сьогоднішній день є платформа dobro.ua [11]. Варто відмітити доволі цікавий підхід до збору коштів на благодійні проєкти. Цей сервіс має декілька скарбничок, що спрямовані на той чи інший напрямок благодійності, зокрема культура та спорт, здоров’я, екологія, соціальна допомога, тощо. Люди можуть поповнити ту чи іншу скарбничку і потім платформа вже на свій розсуд допомагає тому чи іншому благодійному проєкту відповідної категорії. По кожному проєкту є звітність. Крім того, варто зазначити,

що також по кожній скарбничці є статистика, яка показує скільки грошей було витрачено, скільки наразі в скарбничці є та хто останній пожертвував кошти саме у цю категорію. Крім того ще реалізовано можливість фінансово допомогти певному благодійному проєкту. Дуже зручно реалізовано систему пошуку та сортування, доволі приємний та інтуїтивно зрозумілий інтерфейс. Ще один цікавий спосіб благодійності, реалізований на вебплатформі, це SMS благодійність. Відправляючи SMS повідомлення за певним переліком номерів можна пожертвувати невелику суму коштів (приблизно 20 грн.) на конкретну категорію. Слід відмітити і доволі детальне формування звітності по кожному проєкту, де розписано всі випрати, додано необхідні документи, квитанції, фото і текст, тощо. Це допомагає користувачу бути впевненим у тому, що кошти дійсно відправлено на благодійність.

Використання браузерного розширення Wappalyzer [12] дозволило подивитися, які саме технології використано для створення цього сервісу. Вебзастосунок розроблено на мові програмування Python, з використання фреймворку Django. Крім того, з аналізу видно, що платформа підключена до AWS та використовує доволі велику кількість сервісів для аналітики, такі як Hotjar, Google Analytics, Facebook Pixel, тощо.

Це одна аналогічна платформа, під назвою RAZOM [13]. На відміну від попередньої вебплатформи, що дозволяла відправляти грошову допомогу або на певній проект або в загальну скарбничку певної категорії, на цьому сервісі є лише 5 проєктів, розділених між собою, як категорії, тобто конкретного розбиття на певні збори відсутні. Є блок з новинами, опубліковані лише річні звіти. Варто акцентувати увагу на присутність блоку інформації для біженців з великою кількістю порад по еміграції, роботі та навчанні.

За інформацією, отриманою за допомогою браузерного розширення, вказаного раніше, вебсервіс розроблено за допомогою WordPress, використовує велику кількість бібліотек JavaScript та мову програмування PHP. Крім того, для організації БД використано MySQL. До того ж, варто підкреслити використання 3

UI фреймворків, таких як Animate.css, Bootstrap та ZURB Foundation. Але не дивлячись на таку різноманітність використаних технологій, платформа виконую незначну функціональну частину, на відміну від попереднього прикладу.

1.3 Постановка задачі

Зростання агресії з боку російських військ у ході захоплення нашої держави підкреслює важливість вдосконалення системи волонтерської діяльності та благодійних організацій. Це необхідно для ефективного забезпечення необхідними ресурсами наших військових і цивільних, які постраждали від дій агресора, що безперечно підтверджує актуальність даної теми.

Метою проєкту даної кваліфікаційної роботи є покращення системи координації роботи волонтерів та благодійних організацій, контроль та прозорість грошових зборів на допомогу різного типу класифікації та забезпечення швидкої якісної комунікації та обміну ресурсів, шляхом розробки вебзастосунку.

Об’єктом дослідження є система управління волонтерськими ресурсами та взаємодії з благодійними організаціями, її архітектура, функціональні можливості, алгоритми обробки даних, способи інтеграції з іншими сервісами.

Предметом дослідження є набір технологій та методів розробки вебсистеми, яка дозволяє ефективно керувати волонтерськими ресурсами, взаємодіяти з благодійними організаціями, автоматизувати процеси збору даних, забезпечити комунікацію.

Проєкт має задовольняти певні функціональні вимоги, до яких входить:

– можливість реєстрації користувачів (або входу вже зареєстрованих користувачів до системи);

– інтеграція з API (платіжна система);

– можливість користувачем здійснити платіж через платіжну систему на певний конкретний вибраний користувачем збір;

– можливість ознайомлення зі звітами по кожному збору, після зібрання всієї суми;

– можливість ознайомлення зі звітами по кожному запиту (крім запиту типу «Порада»), після закриття запиту;

– можливість створити власний збір;

– можливість редагувати створених збір;

– можливість створити власний запит на допомогу;

– можливість редагувати створений запит;

– можливість створити звіт по збору чи запиту, та редагувати його;

– можливість видалити збір, у випадку, поки ще ніхто не відправив на цей збір кошти;

– можливість видалити запит, у випадку, поки ніхто ще не пропонував допомоги;

– можливість вибору та зміни фото профілю;

– можливість відгукнутися на запит;

– можливість прийняти або відхилити пропозицію допомоги у запиті;

– можливість додавати інших користувачів у друзі;

– можливість поскаржитися на інших користувачів у разі виявлення неправомірних дій;

– можливість ознайомитися з останніми новинами у сфері волонтерської діяльності;

– можливість відправити відгук або звернутися до модераторів проекту з будь яких питань;

– можливість пошуку переліку запитів за ідентифікатором та назвою для швидкого вибору потрібного;

– можливість відсортувати перелік запитів за типом запиту для швидкого вибору потрібного;

– можливість відсортувати перелік запитів за локацією для швидкого вибору потрібного;

– можливість знайти певний збір за ідентифікатором та назвою для

знаходження потрібного;

– можливість відсортувати певний збір за категорією та станом (нові, майже завершені, тощо) для знаходження потрібного;

– можливість ознайомитися з історією проекту та командою розробників;

– можливість стати спонсором проєкту, здійснивши грошовий переказ;

– можливість редагувати власні дані (номер телефону, електрону адресу, тощо);

– можливість видалити власний акаунт (у разі відсутності відкритих зборів та незакритих звітів);

– можливість контактувати з іншими користувачами системи через внутрішній чат платформи;

До нефункціональних вимог проєкту входить:

– весь інтерфейс українською мовою;

– інтуїтивно зрозумілий інтерфейс;

– швидке завантаження сторінок сайту;

– організована система здійснення платежів;

– інтерфейс має бути виконаний у стриманих світлих тонах, а інші деталі, включаючи текст, кнопки, тощо, мають бути у міру яскравими.

Проєкт має охоплювати доволі велику цільову аудиторію, що у свою чергу має включати самостійних волонтерів, тобто тих, хто може допомагати власноруч та не входять у штат співробітників благодійних організацій, волонтерів та представників організацій, інших громадян, які можуть і не бути волонтерами, але мають можливість допомагати фінансово, та ставати спонсорами проєкту. Вікове обмеження зареєстрованих користувачів веб платформи складає 18 років.

Висновки до розділу 1

Проаналізовано предметну область проєкту, що охоплює волонтерську діяльність та роботу благодійних організацій. Визначено основні поняття,

ознайомлено з юридичним аспектом даної тематики.

Проведено дослідження волонтерства та благодійництва сьогоденної України, у якому прийняли участь 60 респондентів. Визначено основні проблеми, з якими стикаються волонтери, потреби людей, що хотіли б доєднатися до даного виду діяльності, рівень довіри громадян благодійним організаціям, мотивації та загальний рівень сприйняття важливості волонтерської роботи. Дане дослідження стало основою для складання технічного завдання проєкту таким чином, щоб всі аспекти були враховані для отримання найкращого результату.

Проаналізовано 5 наукових публікацій зі схожою тематикою і як висновок, можна сказати про те що кожна з публікацій охопила лише окрему складову волонтерства, грошову або організаційну, при цьому технології, що були використані, кардинально відрізняються у кожні з них, що ще раз доводить, що кожна задача має безліч шляхів її вирішення.

Аналіз декількох популярних онлайн сервісів благодійництва дало можливість оцінити і розглянути варіанти реалізації кваліфікаційної роботи. Визначено сильні та слабкі сторони, що також буде враховано під час програмної реалізації проєкту.

У технічному завдані визначено основні функціональні та нефункціональні вимоги, необхідні для реалізації вебзастосунку для координації волонтерської роботи, обміном ресурсів, зручних та безпечних зборів пожертв. Описано цільову аудиторію, що буде користуватися розробленим вебсервісом. Вимоги сформовані таким чином, щоб цільова аудиторія була якомога більшою, щоб залучити більше людей до волонтерською та благодійної діяльності.

2 ТЕХНОЛОГІЇ, МОДЕЛІ І МЕТОДИ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ ВОЛОНТЕРСЬКОГО МЕНЕДЖМЕНТУ

2.1 Методи для вирішення задачі

2.1.1 Методи теорії прийняття рішень

Метод зважених оцінок [13] є одним з найпростіших методів теорії прийняття рішень. Він доволі часто використовується у випадку, коли наявний вибір між декількома рішеннями, що порівнюються між собою за певними критеріями.

Для того, щоб застосувати даний алгоритм, треба оцінити кожне наявне потенційне рішення певною оцінкою за кожним критерієм. Далі треба визначитися, які критерії є найбільш важливими і згідно цього розставити ваги критеріїв. При цьому, сума вагів всіх критеріїв має дорівнювати 1. Після того, як проставлено ваги критеріїв можна обчислити, яке рішення є оптимальним. Для цього необхідно вагу кожного критерія помножити на оцінку рішення за цим критерієм. Потім ці значення сумуються і отримується загальна оцінка рішення. Оптимальним є те рішення, загальна оцінка якого є найбільшою з усіх інших [13].

У **методі простого ранжування** все залежить від пріоритету кожного критерія, за якими порівнюються рішення [14-15]. Найважливіший критерій має пріоритет 1, менш важливий 2, ще менш важливий 3, тощо. Далі розраховуються ваги критеріїв за наступною формулою:

$$w_i = \frac{n - r_i + 1}{\sum_{j=1}^{n}(n - r_j + 1)} \quad (2.1)$$

де w_i – вага критерію i;

n – кількість критеріїв;

r_i – ранг критерію i.

Причому сум ваг всіх критеріїв має дорівнювати 1.

Після цього виконуються дії, описані у методі зважених оцінок, тобто вагу кожного критерія помножити на оцінку рішення за цим критерієм. Потім ці

значення сумуються і отримується загальна оцінка рішення. Оптимальним є те рішення, загальна оцінка якого є найбільшою з усіх інших.

У випадку якщо ранг критерію вказано діапазоном значень, наприклад 1-2 або 1-3, тоді реальний ранг критерію обчислюється як середнє значення меж діапазону, наприклад, якщо ранг вказано як 1-2, тоді буде 1,5.

Пропорційний метод [14] використовує трішки інший підхід у розрахунку вагів критеріїв, на відміну від методу простого ранжування. Вага розраховується з урахуванням певних коефіцієнтів, які надані критеріям. Коефіцієнти показують важливість того чи іншого критерія, тобто чим більше значення коефіцієнту, тим важливіший критерій. Вага кожного критерію розраховується наступним чином: коефіцієнт поточного критерію ділиться на суму всіх коефіцієнтів, після чого оптимальне рішення знаходиться так само як і в методі зважених оцінок.

2.1.2 Алгоритм шифрування даних AES-256 у режимі GCM

AES-256 (Advanced Encryption Standard 256-bit) – алгоритм симетричного шифрування, довжина ключа якого складає 256 біт, до дозволяє забезпечити доволі високий рівень безпеки [16].

Режим GCM [17] – режим шифрування для блокових шифрів, що використовує універсальне хешування над двійковим полем Galois для забезпечення конфіденційності та цілісності.

Алгоритм поєднує у собі відразу 2 функції: шифрування та автентифікацію даних одночасно. Працює алгоритм AES-256-GCM наступним чином:

– генерується секретний ключ на 256 бітів а також вектор ініціалізації IV, що є унікальним для кожного шифрування;

– генерується лічильник;

– IV комбінується зі згенерованим раніше лічильником і шифрується за алгоритмом AES-256, отримується на виході шифротекст;

– використовується хеш-функція для визначення тегу автентифікації, що обчислюється на основі раніше отриманого шифротексту та додаткових

автентифікованих даних AAD;

– отриманий тег допомагає, під час розшифрування, перевірити цілісність даних та автентичність.

Для повноти розуміння даного процесу варто також розглянути безпосередньо сам алгоритм AES-256 [16]. Так як він є симетричним алгоритмом шифрування, то це означає, що той секретний ключ, який використовується при шифруванні, треба використовувати і під час розшифрування. Розмір ключа, 256 біт, робить його стійкішим і безпечнішим, адже такої довжини ключ доволі складно підібрати або вгадати. Так як розмір ключа 256 біт, то й алгоритм має 14 етапів шифрування. Текст, який треба зашифрувати, розбивається на блоки по 16 байтів. Основний секретний ключ розбивається на 14 раундових ключів, кожен з яких використовується на певному етапі шифрування. Кожен з блоків по 16 байт проходить всі 14 етапів, що у собі включають заміну байтів, зсув рядків, змішування стовпців, тощо. Після 14 раундів кожен блок з даними стає шифротекстом, які потім об’єднуються у єдине ціле.

2.2 Технології розробки системи

2.2.1 Вибір мови програмування клієнтської частини вебзастосунку

Серед основних мов програмування клієнтської частини проєкту можна виділити 2: JavaScript [18] та TypeScript [19]. Проведено порівняння цих двох варіантів за певними позиціями, а саме: поріг входу, читабельність коду, гнучкість синтаксису, підтримка у браузерах, швидкість написання коду.

Так, як TypeScript – мова програмування, що була створення у якості надбудови JavaScript, що має більш суворі правила написання коду, то можна казати про те, що JavaScript буде безперечно більш гнучким, поріг входу буде значно нижчим ніж у TypeScript [20]. Крім того, так як браузери не працюють з TypeScript напряму, а компілюють його у JavaScript, тому й говорячи про підтримку у браузерах перевага буде віддана саме JavaScript. Так як TypeScript має статичну

типізацію на відміну від JavaScript, то варто зазначити, що швидкість написання коду буде меншою, хоча дозволяє покращити читабельність коду [21].

Отже, проаналізувавши описані вище особливості цих мов програмування клієнтської частини вебзастосунку, кожному з них поставлено певну оцінку від 1 до 5 (табл. 2.1).

Таблиця 2.1 – Матриця прийняття рішень

Критерій	JavaScript	TypeScript
Поріг входу	5	4
Читабельність коду	4	5
Гнучкість синтаксису	5	4
Підтримка у браузерах	5	4
Швидкість написання коду	5	4

Обрано оптимальне рішення, застосувавши такий метод прийняття рішень, як метод зважених оцінок (табл. 2.2).

Таблиця 2.2 – Застосування методу зважених оцінок

Критерій	Вага	JavaScript	TypeScript	W_JS	W_TS
Поріг входу	0,13	5	4	0,67	0,53
Читабельність коду	0,13	4	5	0,53	0,67
Гнучкість синтаксису	0,30	5	4	1,50	1,20
Підтримка у браузерах	0,30	5	4	1,50	1,20
Швидкість написання коду	0,13	5	4	0,67	0,53
Сума	1,00			4,87	4,13

Отже оптимальним рішенням буде обрати мову програмування саме JavaScript.

2.2.2 Вибір мови програмування серверної частини вебзастосунку

Вибір мови програмування для серверної частини більш розширений, ніж для клієнтської, адже вибирати треба між такими популярними мовами, як PHP [22],

Python [23] та Ruby [24]. Для детального аналізу кожного з них обрано певний перелік критеріїв, до яких входять: доступність хостингу, оптимізація під веб, швидкість розробки, сумісність з MySQL, підтримка поштових функцій, інтеграція з Frontend фреймворками та можливість розширення через сторонні бібліотеки.

Якщо говорити про доступність хостингу, то варто зазначити, що саме у PHP ця доступність найвища серед усіх інших варіантів. Для Python також доволі часто можна зустріти хостинги, але найчастіше всього йому потрібна VPS та хмарна інфраструктура для більш коректної роботи. Що до Ruby, то у випадку його використання хостинг знайти буде набагато складніше, адже йому зазвичай треба VRS та спеціальні платформи.

Говорячи про оптимізацію з веб не можливо не згадати той факт, що PHP з самого початку був створений під веб, тож має вбудовану підтримку всіх необхідних вебфункцій. Python більш універсальна мова програмування, але її можна використовувати для веброзробки, якщо писати на ньому разом з фреймворком Django або Flask. Ruby має доволі гарну оптимізацію під веб, особливо використовуючи фреймворк Ruby on Rails.

Всі три кандидати орієнтовані на швидку розробку, причому кожен через свої властивості: PHP наприклад має велику кількість CMS, Python має простий синтаксис, у випадку використання фреймворку, що пришвидшує написання коду, а Ruby взагалі використовує принцип «конвенції замість конфігурації».

Говорячи про сумісність з MySQL, то варто звернути увагу на те, що всі кандидати знаходяться на одному рівні, адже кожен має свої бібліотеки, що дозволяють комфортно працювати з MySQL.

PHP має заздалегідь вбудовану функцію mail(), що дозволяє швидко працювати з поштовими методами, у Python треба інтегрувати модуль smtplib для виконання такої задачі, тож коду прийдеться дописати більше. Ruby, так само як і PHP, має вбудовану систему для відправки пошти.

Всі визначені вище мови програмування легко інтегруються з Frontend фреймворками, таким як Vue.js чи React, та мають можливість підключення

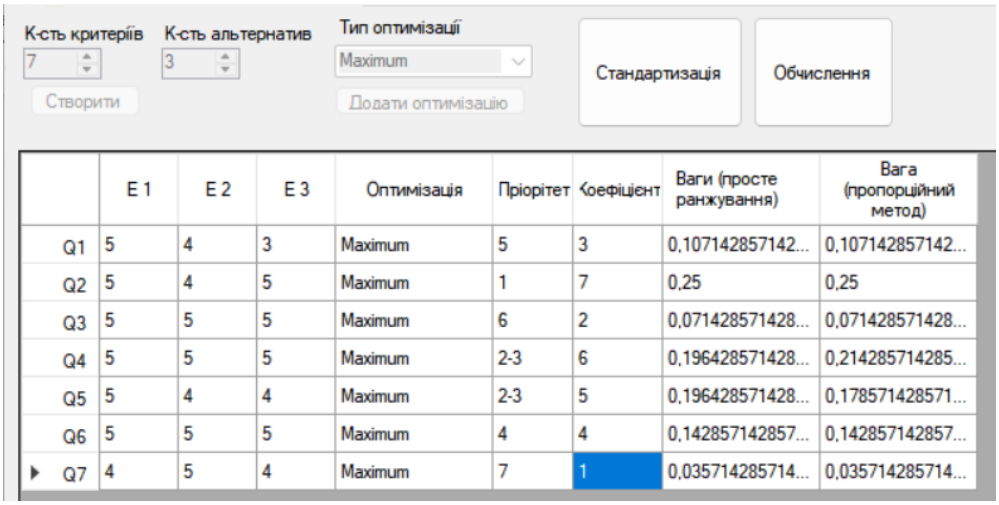
сторонніх бібліотек та модулів, але варто відмітити, що Python, на відміну від інших, має найбільшу екосистему бібліотек, а PHP використовує Composer, що покращує менеджмент залежностей. Ruby ж має RubyGems, менеджер пакунків, де відповідно й зберігається безліч модулів.

Отже, проаналізувавши всі критерії порівняння, можна оцінити кожного з вище описаних кандидатів оцінкою від 1 до 5 наступним чином (табл. 2.3).

Таблиця 2.3 – Оцінки мов програмування для реалізації серверної частини вебзастосунку

Критерій	PHP	Python	Ruby
Доступність хостингу	5	4	3
Оптимізація під веб	5	4	5
Швидкість розробки	5	5	5
Сумісність з MySQL	5	5	5
Підтримка поштових функцій	5	4	4
Інтеграція з Frontend фреймворками	5	5	5
Можливість розширення через сторонні бібліотеки	4	5	4

Застосувавши пропорційний метод та метод простого ранжування з теорії прийняття рішень, то можна побачити результат (рис. 2.1, 2.2).



	E 1	E 2	E 3	Оптимізація	Пріорітет	Коефіцієнт	Ваги (просте ранжування)	Вага (пропорційний метод)
Q1	5	4	3	Maximum	5	3	0,107142857142...	0,107142857142...
Q2	5	4	5	Maximum	1	7	0,25	0,25
Q3	5	5	5	Maximum	6	2	0,071428571428...	0,071428571428...
Q4	5	5	5	Maximum	2-3	6	0,196428571428...	0,214285714285...
Q5	5	4	4	Maximum	2-3	5	0,196428571428...	0,178571428571...
Q6	5	5	5	Maximum	4	4	0,142857142857...	0,142857142857...
Q7	4	5	4	Maximum	7	1	0,035714285714...	0,035714285714...

Рисунок 2.1 – Визначення ваг критеріїв для методу простого ранжування та пропорційного методу

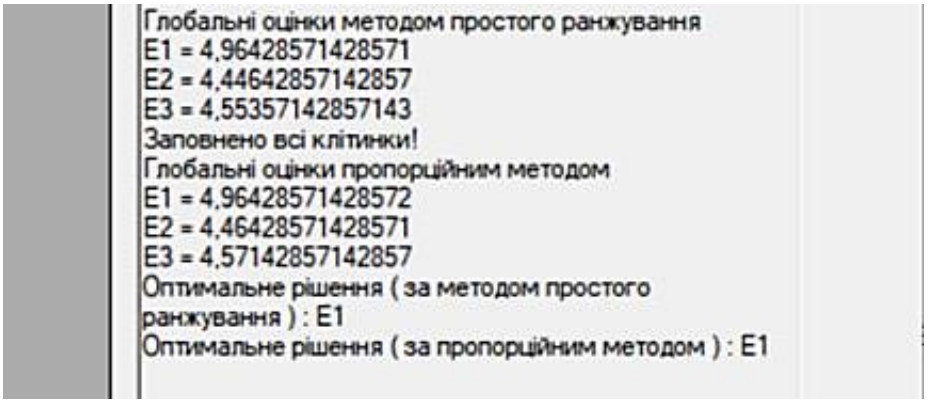


Рисунок 2.2 – Визначення оптимального рішення з варіантів мов програмування

Найбільш оптимальним рішенням є PHP, тож для розробки буде використовуватися саме він.

2.2.3 Вибір фреймворку

Для реалізації клієнтської частини є можливість вибору серед таких фреймворків, як Vue.js [25] та React [26]. Вони є найпопулярнішими на сьогоднішній день. Застосувавши теорію прийняття рішень і проведено порівняння цих фреймворків за такими критеріями: швидкість розробки, документація, поріг входу (простота вивчення), інтеграція з PHP, популярність, робота з формами та валідацією, швидкість та продуктивність, придатність для малих команд розробників, читабельність коду та гнучкість.

Перший критерій для порівняння – швидкість розробки. Так як розробка проєкту кваліфікаційної роботи має доволі жорсткі дедлайни, тобто дається мало часу на реалізацію такого великого застосунку, то цей критерій має доволі важливе місце у списку пріоритетності. Vue.js у свою чергу має інтуїтивний синтаксис, мінімум конфігурації, швидко розгортається, має офіційний CLI, шаблони для старту, на відміну від React, який потребує більших налаштувань, складніший синтаксис JSX, підключення великої кількості бібліотек для повноцінної роботи застосунку [27, 28].

До цього ж можна і віднести простоту навчання, через простоту синтаксису Vue.js та великої кількості вже вбудованих функцій, наприклад router та сховище стану, він значно виграє у React, де все треба підключати вручну, тобто потребує вивчення додаткових бібліотек.

До того ж, та сама простота синтаксису Vue.js дає перевагу у таких критеріях як придатність для малих команд розробників та читабельність коду, адже, порівняно з React, де синтаксис складніший, коду треба писати більше, більше налаштовувати, то й потрібно більше людей для пришвидшення розробки. Крім того, код на Vue.js, за допомогою своєї структурованості та компактності, є більш читабельним, що дає перевагу у випадку подальшого оновлення та покращення застосунку.

Документація що у Vue.js, що у React доволі деталізована, має локалізацію, тобто переведена багатьма мовами. Все розділено за темами, тож легко знайти необхідну інформацію. Окремим позитивним нюансом на користь Vue.js є можливість у документації перемикатися між Options API та Composition API, то змінює приклади коду в залежності від обраного варіанту [25-27].

Що до популярності цих фреймворків не можна сказати, що один з них більш популярний, а інший ні, хочу можна помітити тенденцію росту кількості розробників та компаній, що починають переходити на Vue.js після виходу Vue3. Не дивлячись на те, що раніше React був більш популярним, на сьогоднішній день можна сказати, що вони знаходяться більш-менш на одному рівні [27].

Якщо порівнювати Vue.js та React з точки зору гнучкості, то варто зазначити, що саме React є максимально гнучким, до будь якого завдання знайдеться багато шляхів реалізації. Vue.js, у свою чергу має уніфіковану інфраструктуру, тобто відразу надає рішення без додаткового обрання інструментів [28].

Обидва фреймворки використовують віртуальний DOM для відображення вебсторінок, що дає їм можливість не мати великих відривів у швидкості та продуктивності. Однак, якщо говорити про створення та оновлення компонентів, то Vue.js буде швидший, за рахунок того, що він автоматично оновлює компоненти,

на відміну від React, у якому треба компонент оптимізувати, якщо якісь зміни були внесені, що може призвести до виникнення проблем у випадку, коли проєкт на React треба масштабувати [27, 28].

Через те, що проєкт передбачає постійну роботу з формами, то критерій «робота з формами та валідацією», посідає далеко не останнє місце. Vue.js підтримує двостороннє зв’язування через використання v-model, легкі рішення для валідації, наприклад VeeValidate, а також доволі просте оновлення значень форми, у той час як React має односпрямований data flow, тобто обробку треба прописувати самостійно, адже оновлення відбувається таким чином: с початку оновлюється state а далі перемальовуються дані [25, 27].

Так як в одному з минулих підрозділів, проаналізувавши мови програмування для серверної сторони проєкту, обрано PHP, то можливість інтеграції з цією мовою є актуальною. Обидва фреймворки однаково мають прекрасну інтеграцію з PHP, але, знову ж таки, React потребує для цього більше налаштувань.

Проаналізувавши кожен критерій оцінено кожен фреймворк від 1 до 5 (табл. 2.4).

Таблиця 2.4 – Оцінювання Vue.js та React за певними критеріями (матриця прийняття рішень)

Критерій	Vue.js	React
Швидкість розробки	5	4
Документація	5	3
Поріг входу	5	4
Інтеграція з PHP	5	4
Популярність	4	5
Робота з формами та валідацією	5	4
Швидкість та продуктивність	5	5
Придатність для малих команд розробників	5	4
Читабельність коду	5	4
Гнучкість	3	5

Тепер, застосувавши метод простого ранжування та пропорційний метод, можна обрати оптимальне рішення, згідно даних оцінок цих двох фреймворків за 10 критеріями, використавши ту ж саму програму, відображену на рисунку 2.1. Для цього створено та використано програмний застосунок на мові програмування C#, програмний код якого представлений у додатку А.

2.2.4 Вибір середовища розробки

Для комфортної та якісної розробки доволі важливо обрати кращій варіант середовища розроби. Серед популярніших варіантів сьогодення варто підкреслити такі IDE, як Visual Studio Code [29], WebStorm (JetBrains)[30], та редактори коду Sublime Text [31] та Notepad++ [32].

Кожен з цих варіантів детально проаналізовано та порівняно за певним критеріями: наявність безкоштовної версії (ціна), швидкість та стабільність, можливість легко налаштовувати проєкт, зручність та інтуїтивність інтерфейсу, дебагінг, можливість підключення плагінів та розширення, підтримка технологій, необхідних для розробки, можливість автодоповнення коду, наявність підказок, наявність інтеграції з Git та підтримка Live Preview.

Якщо розглядати вартість вище зазначених IDE та редакторів коду, то можна сказати про те, що всі вони безкоштовні, якщо використовувати у некомерційних цілях.

Найважливішим критерієм вибору є підтримка необхідних обраних технологій для розробки програмного продукту, а саме JavaScript, PHP, Vue.js, HTML, CSS (SCSS). Visual Studio Code та WebStorm підтримують усе з вищезазначеного. Sublime Text має базову підтримку цих технологій з можливістю розширення через плагіни, а ось Notepad++ підійде лише для невеликого коду, що не дуже підходить для поточного проєкту.

Якщо говорити про зручність та інтуїтивність інтерфейсу середовища розробки, то варто зазначити, що кожний варіант є доволі комфортним по мірі можливостей програмного продукту, хочу у випадку поточного вебзастосунку,

коли потрібно мати всі необхідні функції під рукою, то комфортнішими варіантами будуть саме Visual Studio Code та WebStorm, адже не потребують великої кількості маніпуляції для знаходження того чи іншого інструменту. Дивлячись з власних переконань, Visual Studio Code є найбільш оптимальним у цьому плані.

Для будь якого розробника важливо мати можливість легко налаштовувати проєкт. У випадку з Sublime Text та Notepad++ проєкт треба налаштовувати в ручну і вже потім відкривати у редакторі коду. У випадку з WebStorm це все автоматизовано. Visual Studio Code надає таку можливість, доволі зручно з CLI та підключеними плагінами.

Visual Studio Code та WebStorm відрізняються своєю стабільністю та швидкістю. Перший з них легкий, більшість чого можна за потреби налаштовувати та встановлювати, тож він легкий, тому й швидкий, WebStorm важчий, адже має багато вбудованих можливостей, але не відстає по швидкості від Visual Studio Code. Sublime Text та Notepad++ легкі, тому і швидкі, хочу на власному досвіді бували моменти, коли під час роботи вони зависали.

У Visual Studio Code, WebStorm, та у Sublime Text є можливість підключати плагіни та розширення, причому Visual Studio Code має дуже велику бібліотеку розширень, у WebStorm багато вбудованих бібліотек, на відміну від Sublime Text, де перелік розширень значно менший ніж у попередніх кандидатів. Notepad++ має можливість підключення плагінів, хоча й незначну кількість.

Розглядаючи тему можливості зупинки виконання, перегляду змінних для пошуку помилок, тобто дебагінг, то WebStorm має потужний вбудований дебагер, у Visual Studio Code його можна налаштувати під конкретні потреби, у Sublime Text та Notepad++ такого немає.

Що до автодоповнення та підказок, то з цією задачею успішно справляються Visual Studio Code та WebStorm. Sublime Text також має функцію автоматичного допису коду, але доволі обмежену. Якщо говорити про Notepad++, то варто зазначити, що там є підказки, які з’являються списком при написані коду, але повноцінного автодоповнення немає.

Варто зазначити важливість середовища розробки мати наявність інтеграції з Git. WebStorm має повну інтеграцію, так само й у Visual Studio Code є панель для використання Git відразу під час роботи над проєктом. Що до Sublime Text та Notepad++, то інтеграція з Git там відсутня, хоча для Sublime Text можна поставити відповідний плагін.

Підтримка Live Preview буде доволі корисна при розробці вебзастосунку, тому цей критерій має не останню вагу у переліку. У WebStorm ця функція вбудована, у Visual Studio Code можна налаштування за допомогою розширення Live Server, для Sublime Text можна використати плагіни для підключення, а ось у Notepad++ цього зробити не можна.

Проаналізувавши всі вище зазначені моменти, оцінено кожне середовище розробки оцінкою від 1 до 5 по кожному критерію (табл. 2.5).

Таблиця 2.5 – Оцінювання Visual Studio Code, WebStorm, редактори коду Sublime Text та Notepad++ за певними критеріями

Критерій	Visual Studio Code	WebStorm	Sublime Text	Notepad++
Ціна	5	5	5	5
Швидкість та стабільність	5	4	5	5
Можливість легко налаштовувати проєкт	4	5	1	1
Зручність та інтуїтивність інтерфейсу	5	5	5	5
Дебагінг	5	5	2	1
Можливість підключення плагінів та розширення	5	5	3	2
Підтримка технологій, необхідних для розробки	5	5	3	2
Автодоповнення коду, наявність підказок	5	5	4	2
Наявність інтеграції з Git	5	5	3	1
Підтримка Live Preview	5	5	3	1

Скориставшись вже продемонстрованим вище програмним застосунком, програмний код якого зазначено у додатку А, та такими методами прийняття рішень як пропорційний метод та метод простого ранжування можна підсумувати, що у даному випадку присутні 2 оптимальних рішення, тоді, керуючись власним вподобаннями, у розробці буде використовуватися Visual Studio Code.

2.2.5 Вибір сервісу для прототипування

Вибір якісного сервісу для прототипування дозволить пришвидшити процес розробки, моделювання та аналізу сценаріїв роботи вебсервісу. Серед найпопулярніших варіантів таких сервісів є Figma та Adobe XD [33].

Кожен з цих варіантів детально проаналізовано та порівняно за певним критеріями: різноманітність типів платформи, ціна, можливість створення інтерактивних переходів та анімацій, наявність плагінів та інтеграцій, платформа спільного доступу, можливість працювати у команді в реальному часі та швидкість роботи.

Якщо говорити про Figma, то варто зазначити, що вона є і як вебзастосунок (можна працювати у браузері), і як десктоп-застосунок, тобто можна встановити на ПК. Цей сервіс є безкоштовним, але є деякі невеликі обмеження, які можна прибрати, придбавши підписку. У Figma можна створювати величезну кількість переходів, анімації. Компоненти дуже гнучкі, що дозволяють реалізувати будь які дизайнерські рішення. Figma має величезну бібліотеку плагінів, наприклад, для обробки зображень, що значно пришвидшує прототипування, адже не має потреби у застосуванні редакторів фото, наприклад того ж самого Photoshop. Цей сервіс дозволяє працювати у команді, для цього просто треба надіслати посилання на проєкт іншим учасникам команди. Через те, що Figma залежить від інтернет з’єднання він може працювати повільно, хоча практично, використовуючи встановлену версію на ПК, затримок у швидкості не помічено, але видає повідомлення про те, що зміни, внесені у проєкт не зберігаються без інтернет з’єднання.

З іншої сторони, не можливо не згадати про Adobe XD. Порівняно з Figma він дещо програє, адже цей сервіс не має вебверсії, тобто треба встановлювати на комп’ютер. Він, як і Figma, є безкоштовним, хоча має більше обмежень на безкоштовній версії, можливо оформити підписку. Якщо згадати про анімації та переходи, то у Adobe XD варіантів менше, ніж у Figma, як і плагінів. Можна працювати над одним проєктом командою, але треба мати обліковий запис Adobe. Швидкість роботи трішки більша ніж у Figma за рахунок того, що Figma працює через хмару і залежить від інтернет з’єднання.

Проаналізувавши всі основні критерії проставлено наступні оцінки (табл. 2.6):

Таблиця 2.6 – Оцінювання Figma Adobe XD за певними критеріями

Критерій	Figma	Adobe XD
Різноманітність типів платформи	5	4
Ціна	5	4
Можливість створення інтерактивних переходів та анімацій	5	4
Наявність плагінів та інтеграцій	5	4
Платформа спільного доступу	5	4
Можливість працювати у команді в реальному часі	5	5
Швидкість роботи	4	5

Використано програмний застосунок, програмний код якого зазначено у додатку А, та такими методами прийняття рішень як пропорційний метод та метод простого ранжування знайдено оптимальне рішення, яким є Figma.

Висновки до розділу 2

Проаналізовано та детально вивчено основні методи, що використані під час виконання кваліфікаційної роботи, серед яких важливо виділити алгоритм шифрування даних AES-256 у режимі GCM, методи теорії прийняття рішень, такі як пропорційний метод, метод простого ранжування та метод зважених оцінок.

Для обрання технологій, що будуть використані під час розробки

вебзастосунку, застосовано такі методи прийняття рішення, як метод зважених оцінок, метод простого ранжування та пропорційний метод. У результаті, прийнято рішення використання мови програмування JavaScript для клієнтської частини застосунку, мову програмування PHP для реалізації роботи з сервером та БД, фреймворку Vue.js та середовища розробки Visual Studio Code.

Обрані методи та технології дозволять швидко і якісно створити вебсервіс, адже вони мають достатньо переваг, порівняно з іншими варіантам, описаними у розділі.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис вхідних даних

Для реалізації поставленої цілі, а саме створення вебзастосунку для управління волонтерськими ресурсами та взаємодії з благодійними організаціями, виокремлено декілька основних сутностей, реалізація яких забезпечить цілісну роботу системи. До таких сутностей відносяться наступні: спонсор (Sponsor), розробник (Developer), користувач (User), новина (New), збір (MonetaryFee), звіт (Report), запит (Request), чат (Chat) та повідомлення (Messages).

Сутність Спонсор (Sponsor) необхідна для візуалізації на декількох сторінках, так як спонсорами є компанії, а відповідно, їм маємо дякувати за фінансування проєкту. Ця сутність має такі поля, як ідентифікатор, назва компанії та посилання на логотип компанії.

Також виокремлено розробників (Developer). Для збереження та використання інформації про них використано такі поля, як: унікальний ідентифікатор, ім’я, цитата та посилання на фотографію.

Найважливішою сутністю є Користувач (User). Для нього виділено такі поля, які зберігають дані, а саме: унікальний ідентифікатор, логін, пароль, електронна пошта, номер телефону, ім'я, прізвище, по-батькові, дата народження, номер паспорту, дата реєстрації, посилання на фото, організація (якщо користувач працює у благодійній організації), позиція (або самостійний волонтер або волонтер при організації), скільки користувач допомагав закривати збори та скільки допомагав з запитами.

Логін та пароль необхідні для входу у систему, електронна пошта – для відправки повідомлень у технічну підтримку та отримання повідомлень від системи, номер телефону для зв’язку, дата народження для перевірки віку користувача, номер паспорту для ідентифікація користувача, а всі інші дані необхідні для статистики.

Сутність Новина (New) має наступні поля: унікальний ідентифікатор, заголовок, контент (основний текст новини), посилання на фотографію новини, посилання на картинку (обкладинку) та дату розміщення. Новина не пов’язана з користувачем, так як користувачі не будуть створювати чи взаємодіяти з цією сутністю, на відміну від наступних.

Збір (MonetaryFee) – один з важливих елементів системи, так як він зберігає всю важливу інформацію про збір, а саме унікальний номер, тип, заголовок, вміст (основний текст), посилання на фото збору, посилання на обкладинку, IBAN, ціль (кількість коштів, яку треба зібрати), кількість зібраних коштів, ідентифікатор користувача, який створив збір.

Не менш важливим є Запит (Request). Він, в свою чергу, має такі поля: унікальний номер, тип, локація, заголовок, основний текст запиту, посилання на фото, посилання на обкладинку, ідентифікатор користувача, який створив запит, кількість, яку треба зібрати (якщо тип запиту «Матеріали»), кількість, яку зібрали, та міра виміру, наприклад кілограми, літри, тощо. Тип запиту може бути одним з 3 варіантів, а саме «Матеріали», «Фіз. допомога» та «Порада», тому останні 3 поля можуть мати пусте значення. Локація для запиту необхідна у тому випадку, якщо користувачі захочуть знайти запити у своєму місті.

Звіт (Report) – невід’ємна частина системи. Такі поля, як основний текст звіту, статус (відправлено звіт чи ні), посилання на фото звіту, посилання на картинку (обкладинку), ідентифікатор запиту (може бути пустим, якщо звіт не відноситься до запиту) ідентифікатор збору (може бути пустим, якщо звіт відноситься до запиту, а не до збору). Звіт – доволі важлива сутність, адже саме вона забезпечує прозорість роботи волонтерів та має підвищити рівень довіри інших громадян.

Останніми двома елементами є Чат (Chat) та Повідомлення (Message). Саме вони забезпечать можливість обміну повідомленнями між користувачами. Клас Чат зберігає інформація про користувачів, які спілкуються, тобто ідентифікатор першого користувача та другого, а також ідентифікатор останнього повідомлення, який необхідний для того, щоб можна було у списку чатів відразу побачити останнє,

що було у чаті. У свою чергу Повідомлення зберігає у собі ідентифікатор користувача, який відправив повідомлення, ідентифікатор особи, яка отримує це саме повідомлення, текст повідомлення, дата та час відправлення, а також статус («нове» чи «прочитано»).

Зв’язок між цими елементами системи дозволить швидко знаходити необхідну інформацію для візуалізації її у проєкті.

3.2 Опис структури системи

Як і кожен вебзастосунок, поточна вебсистема складається з певної кількості вебсторінок. Вебзастосунок для управління волонтерськими ресурсами та взаємодії з благодійними організаціями складається з таких сторінок, зазначених на схемі у додатку В.

Кожна сторінка містить меню, що складається з логотипу, кнопки переходів на інші сторінки, кнопку відкриття спливаючого вікна зі списком підрозділів профілю та посиланням на саму сторінку профілю, та кнопку відкриття спливаючого вікна з чатами поточного користувача з іншими, з якого, при натисканні на чат, відкриється спливаюче вікно конкретного чату.

Також, варто зазначити, що кожна сторінка має футер, розміщений у самому кінці. Винятками сторінок, де футер відсутній, є сторінка реєстрації та сторінка авторизації користувача.

Головна сторінка вебплатформи, у свою чергу складається з декількох основних блоків, таких як:

– головний банер, що містить заклик приєднатися до платформи, кнопки для швидкого переходу на сторінку реєстрації та на сторінку зборів;

– блок «Про нас», де стисло описано проєкт, його цілі та мета створення, а також наявна кнопка переходу безпосередньо на сторінку «Про нас»;

– блок «Лідери», де за задумом мають бути найактивніші волонтери платформи, але наразі блок виконує лише стильову та візуальну функцію;

– блок «Залишилося ще трішки» де розміщено 3 збори, відсоткове значення закриття яких найбільше за інші збори;

– блок «Долучайся до змін», що містить у собі кнопку для швидкого переходу на сторінку реєстрації користувачів та мотивуючий текст;

– блок «Останні новини», що містить 3 останні новини, опубліковані на сайті;

– блок «Спонсори», що висвітлюють логотипи всіх компаній, які підтримують проєкт, а також мотивують користувача самому стати спонсором та фінансово підтримати розвиток проєкту.

Сторінка «Про нас» висвітлює інформацію про проєкт, а саме описує цілі, мету проєкту, історію створення, тощо. Сторінка складається з анімованого логотипу вебзастосунку, інформацією, карток розробників, у яких розказується про їх задуми створення поточного проєкту, та блоку «Спонсори», описаному раніше.

Сторінка зборів також складається з багатьох важливих деталей, а саме: анімований заклик-мотивація допомагати людям, які цього потребують, поле пошуку певного збору, поле для сортування зборів за певними ознаками, наприклад найновіші, майже закритті, тощо, поле для фільтрації зборів за певною категорією, список карток зборів, що динамічно змінюється залежно від заданих фільтрів. Картка збору має стислу інформацію про збір, до якої належить назва збору, ціль, яку збирають, стислий опис, кількість відсотків закриття збору.

Сторінка кожного конкретного збору містить повну інформацію про збір, статистику закриття збору та можливість долучитися і здійснити грошовий переказ на цей збір.

Сторінка з новинами містить анімований заголовок сторінка та список карток новин, на яких відображена коротка інформація про новину, включно з заголовком та датою викладення. А от вже на сторінці окремої новини можна побачити повну інформацію.

Сторінка «Контакти» описує інформацію для користувачів, куди можна звернутися з певними питаннями чи проблемами. Крім того, на сторінці є форма, для швидкого зв’язку.

Сторінка «Звіти», як і в попередніх випадках, містить анімований заголовок, поле для пошуку певного звіту та перелік карток звітів, які містять коротку інформацію. Сторінка певного звіту містить повну інформацію, а також додатково інформацію про той збір чи запит, до він прикріплений.

Сторінка «Запити» містить перелік карток запитів з короткою інформацією про кожний запит, поле пошуку, поле фільтрації за типом запиту та поле фільтрації за локацією. Повну інформацію про конкретний запит та способи взаємодії можна побачити на його сторінці.

Сторінка авторизації користувача складається з невеличкої форми для вводу логіну та пароль, кнопки, що дозволить зайти на сайт не реєструвавшись у системі (функціонал системи для такого користувача буде обмежено),а також кнопки, яка переведе користувача на сторінку реєстрації.

Сторінка реєстрації має форму для введення всіх необхідних даних користувача, описаних у попередньому розділі.

Сторінка профілю поточного користувача є однією з найбільш обширних, адже вона складається з декількох підсторінок і має власне меню. До підсторінок варто віднести наступні:

– сторінка «Профіль», яка містить всю інформацію про поточного користувача;

– підсторінка «Друзі», що містить перелік друзів поточного користувача з можливістю видалення необхідних з них або відкриття чату з конкретним другом;

– підсторінка «Налаштування», яка дозволяє змінити деякі дані користувача, які є не критично важливими на платформі;

– підсторінка «Збори», що містить перелік зборів, створених користувачем;

– підсторінка «Запити», яка має у своєму складі список запитів, створених поточним користувачем.

Сторінка профілю іншого користувача описує основну інформацію про нього та має у своєму складі функціональні кнопки для взаємодії, наприклад «Додати у друзі», «Написати», тощо.

3.3 Проєктування

Проєктування э одним з найважливіших та фундаментальних етапів розробки програмного продукту. Відповідно до вхідних даних створено, налаштовано та заповнено БД з використанням потужного веб інтерфейсу для управління БД phpMyAdmin (рис. 3.1).

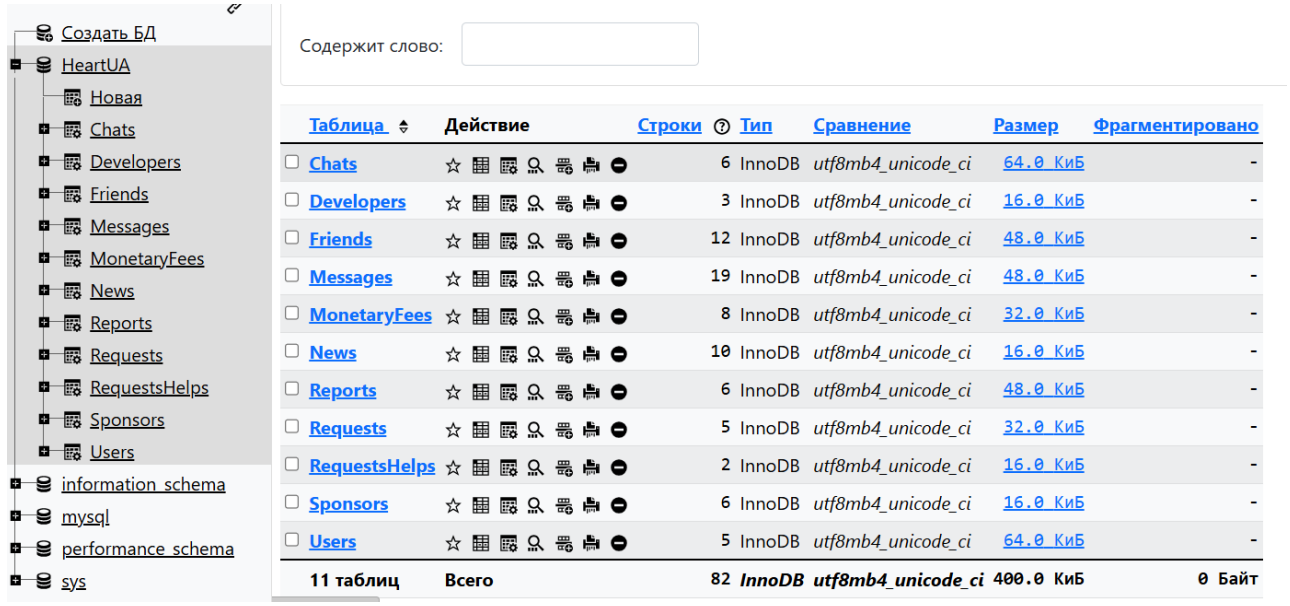


Рисунок 3.1 – Повна структура БД проєкту

Таблиця Chats зберігає інформація про усі чати. Поля таблиці зазначено відповідно до описаних раніше вхідних даних. Поле унікального ідентифікатору є первинним ключем, а отже не може бути пустим, заповнюється автоматично. Поля idUser1 та idUser2 є зовнішніми ключами, що посилаються на унікальні ідентифікатори користувачів, між якими є чат та повідомлення. Останнє поле lastMessage також є зовнішнім ключем та посилається на унікальний ідентифікатор певного повідомлення, що по факту є останнім у чаті між двома користувачами.

Згідно вище зазначеним вимогам до полів таблиці створено та налаштовано необхідні зв’язки (рис. 3.2). Відповідно у результаті вийшли, що зв’язок між полем idUser1 таблиці Chats та полем id таблиці Users є зв’язком багато до одного (N:1), адже один користувач може бути у багатьох чатах. Теж саме стосується і зв’язку поля idUser2 таблиці Chats та поля id таблиці Users. А от зв’язок між полем lastMessage таблиці Chats та полем id таблиці Messages є зв’язком один до одного (1:1), адже поле lastMessage посилається на одне конкретне повідомлення.

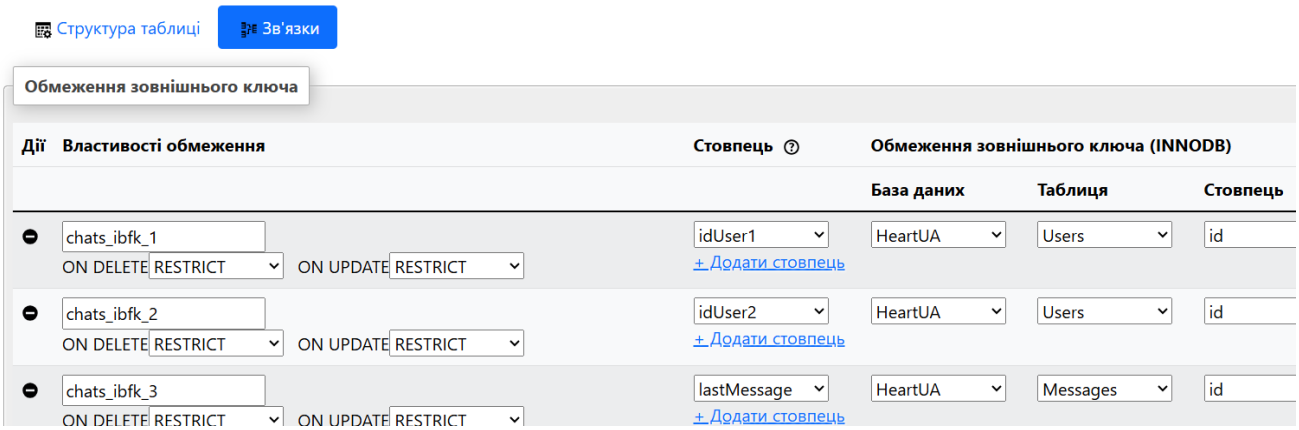


Рисунок 3.2 – Налаштування зв’язків таблиці Chats

Наступна таблиця БД Developers зберігає інформацію про розробників проєкту. Всі поля задані відповідно описаних вище вхідних даних. Таблиця не має зв’язків з іншими таблицями. Поле name зберігає ім’я розробника. Поле quote містить цитату розробника. Поле photo містить відносне посилання на фото розробника, що знаходиться у папці на сервері.

Таблиця Friends містить у собі 3 поля, що несуть інформацію про те, який користувач кого додав у друзі. Поля idUser та idFriend є зовнішніми ключами, що посилаються на індивідуальній ідентифікатор користувача. Ці поля не можуть бути пустими.

Так як у таблиці наявні зовнішні ключі встановлено та налаштовано зв’язки між полями таблиці Friends та таблиці Users. Зв’язок між полем idUser таблиці Friends та полем id таблиці Users є зв’язком багато до одного (N:1), адже один

користувач може дружити з багатьма іншими користувачами. Теж саме стосується і зв’язку поля idFriend таблиці Chats та поля id таблиці Users, адже 1 користувач може бути другом для багатьох інших.

Таблиця Messagers зберігає інформацію по кожному повідомленню з чатів. Структура таблиці створена з урахуванням всіх вхідних параметрів, описаних раніше. Поля idSendUser та idRecipient є зовнішніми ключами, що посилаються на індивідуальній ідентифікатор користувача. Поле idSendUser посилається на користувача, що відправив повідомлення, а поле idRecipient посилається на користувача, який отримав повідомлення. Поле message містить у собі текст повідомлення. Поле dataTime описує дату та час відправлення повідомлення. Поле має формат текстового рядка формату «Y:m:d H:i:s». Останнє поле status містить статус повідомлення, яке може бути або новим, або прочитаним.

Так як таблиця має зовнішні ключі, тож налаштовано необхідні зв’язки між таблицями. Відповідно зв’язок між полем idSendUser таблиці Messages та полем id таблиці Users є зв’язком багато до одного (N:1), адже один користувач може відправляти повідомлення багато кому. Теж саме стосується і зв’язку поля idRecipient таблиці Chats та поля id таблиці Users, адже 1 користувач може приймати повідомлення від великої кількості інших користувачів.

Таблиця MonetaryFees, на відміну від попередніх має у своєму складі набагато більше полів. Вона описує всю інформацію про відкритті та закритті збори. Поля створені опираючись на вказаний перелік вхідних даних сутності Збір. Поле type вказує на тип збору (військовий збір, медичний, для тварин, тощо). Поле title вказує на заголовок збору. Поле content зберігає основний текст. Поля image та picture вказують на фотографію, що буде видно на сторінці збору та обкладинку картки збору відповідно. Поле idUser вказує на користувача, що створив збір. Це поле є зовнішнім ключем. Поле IBAN вказує рахунок для відправки коштів. Поле має фіксовану довжину у 29 символів згідно формату IBAN України. Поле goal описує ціль збору у гривнях. Подібний тип даних має й поле amount, яке описує поточну суму, яку зібрали за цей збір.

Так як поле idUser є зовнішнім ключем, тож треба налаштовувати зв’язки таким самим способом, як і у попередніх прикладах.

Наступна таблиця News, що описує новини платформи, не має зовнішніх ключів і є самостійною. Поле title описує заголовок новини. Поле content вказує на основний текст. Поля image та picture вказують на фотографію, що буде видно на сторінці новини та обкладинку картки новини відповідно. Поле date – дата розміщення новини на платформі, має формат текстового рядка формату «Y:m:d H:i:s».

Таблиця Reports представляє собою структуру, яка зберігає дані про звіти. Поле content вказує на основний текст. Поле isActive зберігає статус звіту (0 – звіт відсутній, 1 – звіт є). Поля image та picture вказують на фотографію, що буде видно на сторінці звіту та обкладинку картки звіту відповідно. Поле isDonate зберігає значення id збору, якщо цей звіт відноситься до нього. За замовчуванням може бути порожнім, якщо наступне поле буде заповнено. Поле isRequest зберігає значення id запиту, якщо цей звіт відноситься до нього. За замовчуванням може бути порожнім, якщо поле idDonate буде заповнено.

Наявність зовнішніх ключів підтверджує необхідність налаштування зв’язків. Зв’язок поля idDonate таблиці Reports з полем id таблиці MonetaryFees є зв’язком один до одного (1:1), тобто вказує на те, що один звіт може відноситися до одного збору. Така сама ситуація і з полем idRequest та полем id таблиці Requests, тобто один звіт може відноситися до одного запиту.

Таблиця Requests зберігає інформацію про запити, містить 11 полів, одне з яких є зовнішнім ключем. Поле type описує тип запиту (матеріали, фізична допомога, порада). Поле title містить заголовок запиту, тип даних якого varchar та максимальний розмір до 40 символів, як і у поля location, що зберігає інформацію про населений пункт. Поле content вказує на основний текст. Поля image та picture вказують на фотографію, що буде видно на сторінці новини та обкладинку картки новини відповідно.

Наявність зовнішніх ключів підтверджує необхідність налаштування зв’язків. Зв’язок поля idUser таблиці Requests з полем id таблиці Users є зв’язком багато до одного (N:1), тобто вказує на те, що один користувач може створити декілька запитів.

Таблиця Sponsors зберігає інформацію про спонсорів проєкту. Поле name відповідає за назву компанії спонсора, а logo зберігає відносне посилання на логотип компанії. Обидва поля є текстовими рядками з обмеженим розміром. Налаштування зв’язків даної таблиці є недоречним через відсутність зовнішніх ключів.

Наступною таблицею БД є таблиця Users, що зберігає інформацію про усіх користувачів системи.

Поле login зберігає логін користувача у вигляді текстового рядка обмеженого розміру. Поле є обов’язковим і не може бути пустим. Крім того, login унікальним альтернативним ключем, тобто декілька однакових значень у табличці бути не може. Поле email має такі самі характеристики, також є унікальним ключем, але зберігає вже електронну пошту користувача.

Поле password так само є обов’язковим текстовим полем, що зберігає пароль, використовуючи який користувач може зайти до системи. Велика кількість виділеного місця під це поле зумовлена подальшим шифруванням та збереження шифротексту.

Поле phone зберігає номер телефону користувача. Розмір цього текстового поля зумовлене кількістю символів у стандартному українському номері з урахуванням, що перші 4 символи це «+380». Поле є унікальним альтернативним ключем, тож може лише 1 раз зберегтися у БД.

Поля name, lname, sname зберігають прізвище, ім’я та по-батькові користувача відповідно. Кожне з них є текстовим полем з обмеженим розміром до 40 символів. Ці поля є обов’язковими та не можуть бути пустими.

Поле birthday – дата народження користувача, формат якого «Y:m:d».

Поле passport є обов’язковим текстовим полем, що зберігає номер паспорту, що служить підтвердженням правильності та актуальності персональних даних користувача. Велика кількість виділеного місця під це поле зумовлена подальшим шифруванням та збереження шифротексту.

Поле image додане для зберігання посилання на фото профілю користувача, тип якого є текстовим рядком обмеженого розміру. Поле не є обов’язковим, фотографію профілю можна буде додати вже після реєстрації.

Поле organization треба у тому випадку, коли користувач відноситься до певної благодійної організації. За замовчуванням поле має значення «none». При реєстрації користувач не вказує організацію, хоча може змінити значення у налаштуваннях. Це поле типу текстовий рядок з обмеженим розміром.

Поле position містить невеликий опис, чи займає користувач якусь позицію у благодійній організації. Якщо ж ні, то за замовчуванням, поле має значення «самостійний волонтер».

Поле dataRegistration заповнюється датою реєстрації користувача у форматі «Y:m:d» і є обов’язковим.

Останні 3 поля, а саме numDonates, numHelpRequest та numFriends необхідні для ведення невеличкої статистики. numDonates зберігає кількість грошових переказів, які здійснив користувач, numHelpRequest – поле для кількості разів, які користувач допомагав закривати запити, а numFriends висвітлює кількість друзів, яких додав до себе користувач.

Таблиця RequestsHelps описує пропозиції у запитах, тобто, наприклад, якщо запит на матеріали, то інший користувач може поділитися якоюсь кількістю, запропонувавши, а вже користувач, що створив запит, може прийняти пропозицію. Таким самим чином працює і для запиту типу «Фіз. допомога», і для поради.

Поле idR зберігає унікальний ідентифікатор запиту, idU – ідентифікатор користувача, що пропонує допомогти закрити запит, count – необхідний для зберігання кількості запропонованого матеріалу, а advice описує текст пораду.

Так, як таблиця має декілька зовнішніх ключів, тож створено необхідні зв’язки. Зв’язок поля idU таблиці RequestsHelps з полем id таблиці Users є зв’язком багато до одного (N:1), тобто вказує на те, що один користувач може допомагати по декільком запитам. Поле idR таблиці RequestsHelps має зв’язок з полем id таблиці Requests типу багато до одного (N:1), адже на один запит може відгукнутися декілька разів.

Загалом структурна схема зв’язків, описаних у даному підрозділі, має вигляд, вказаний у додатку Б.

3.4 Моделювання

Для етапу моделювання використано раніше проаналізований інтерфейс для прототипування Figma.

Спершу, ознайомившись з усіма аспектами волонтерської сфери визначено візуальні потреби у дизайні вебзастосунку. Визначено, що сайт має бути у світлих відтінках, мати ненав’язливий і простий вигляд, вміщувати у себе елементи формальності та патріотизму.

Відтінки кольорів обрано згідно тематики та власних вподобань, оформлено у вигляді демонстраційного об’єкту та додано до групи «Стилі кольорів» у Figma для подальшого можливого редагування або зміни певного кольору у всьому документі одночасно (рис. 3.3). Кожний колір призначений для оформлення певних елементів, наприклад фону, тексту або кнопок, але для введення різноманітності у проєкті деякі елементи можуть використати кольори не за їх безпосереднім призначенням. Прикладом таких є кнопки, колір яких по теорії має бути жовтим, але для деяких карток кнопки створено синіми, а текст світло-сірим для покращення читабельності сайту.

Рисунок 3.3 – Демонстрація кольорової гами проєкту

Після обрання кольорової гами, обрано шрифти для кожного з типів тексту, таких як основний текст, заголовки різних типів, кнопок меню, тощо (рис. 3.4).

Для заголовку першого рівня обрано шрифт Lora, розміром 48 пт, накреслення напівжирне. Такий шрифт точно приверне на себе увагу, тож його застосовано у найважливіших заголовках вебплатформи. Для заголовку другого рівня підібрано шрифт Monserrat, розміром 36 пт, напівжирне накреслення, а для третього рівня – той самий Monserrat, тільки розмір зменшено до 28 пт.

Основний текст оформлено шрифтом Open Sans, розміром 18 пт та звичайним накресленням. У деяких випадках, для підкреслення певної інформації в основному тексті використано напівжирне накреслення, тож напівжирний основний текст також оформлено окремою опцією для зручності.

Для невеликих карток, у яких треба вмістити більшу частину основної інформації оформлення основного тексту, згаданий вище, є неналежним і завеликим, тож окремою опцією створено зменшений варіант оформлення основного тексту зі зменшенням у розмірі шрифту до 14 пт.

Для заголовку головної сторінки обрано окремий шрифт Playfair Display, розмір для шрифту заданий як 56 пт., накреслення напівжирне.

Для кнопок меню обрано шрифт Ubuntu за рахунок своєї простоти та серйозності, розмір якого задано 18 пт.

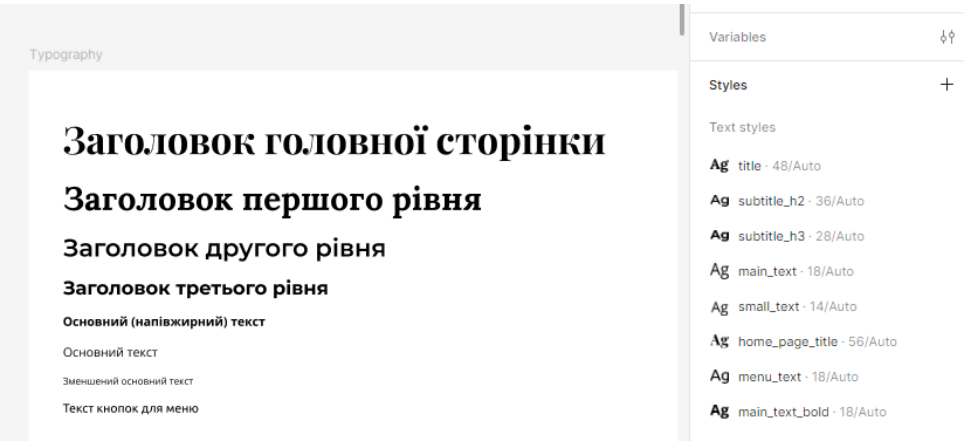


Рисунок 3.4 – Демонстраційний варіант обраних шрифтів для кожного виду тексту на сайті

Наступним і доволі важливим кроком виконано створення логотипу. Використавши вебсервіс Canva, підібрані шрифти та кольори вийшло відтворити логотип, який своїм виглядом натякає користувачу на тематику вебплатформи (рис. 3.5). У якості головного елементу логотипу обрано квітку. Логотип оформлений у відтінках державного прапору України, натякаючи на квітучість та розвиток країни.

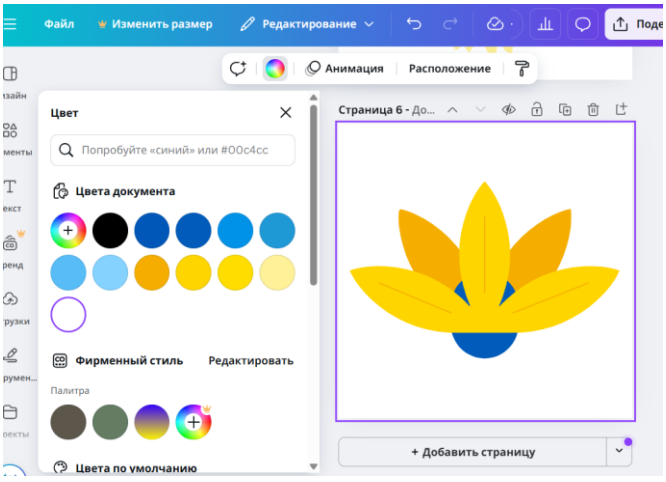


Рисунок 3.5 – Створення логотипу вебзастосунку

Для прийняття рішення що до назви вебплатформи проаналізовано багато різних варіантів як українською так і англійською мовою. У якості ключових слів застосовані наступні: любов, добро, допомога, серце, волонтерство, разом, поруч,

тощо. В результаті обрано назву HeartUA, яка вміщає у собі слово «серце» англійською мовою, що асоціюється з добром та чуйністю, та символьний код країни.

Відповідно до структури системи, описаної декількома підрозділами раніше, створено дизайн документ. Перша сторінка яку бачить користувач при вході на платформу – сторінка з формою авторизації (рис. 3.6). На ній відображено логотип з назвою вебзастосунку, 2 поля для вводу даних, необхідних для входу та дві кнопки. Перша переводить користувача на головну сторінку (у випадку правильності введення логіну та паролю), а інша – на сторінку реєстрації. Кнопку та поле для вводу тексту розроблено як компоненти, що у свою чергу дозволяє використовувати ці компоненти багато разів у дизайні, при цьому змінюючи їх мінімальні налаштування, наприклад текст, чи колір. Додатково відображено посилання «Увійти як гість» , що дозволяє людям відвідати сайт без реєстрації, хоча більшість функціоналу, пов’язаного з профілем, чатами, тощо, буде недоступно.

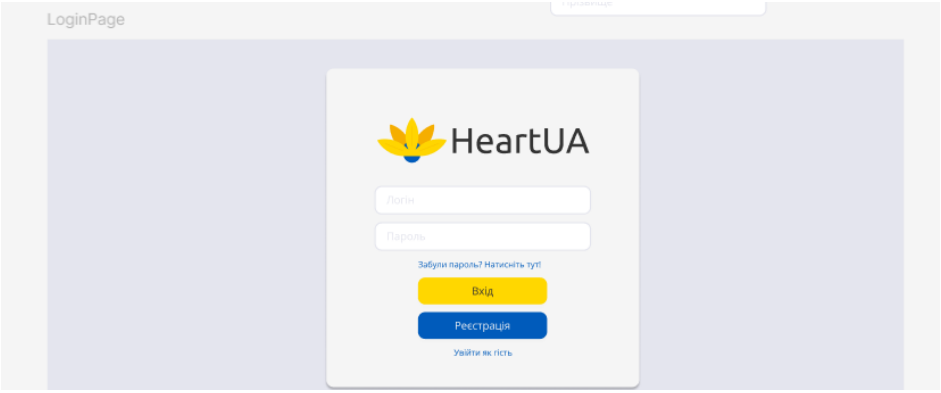


Рисунок 3.6 – Дизайн сторінки авторизації користувача

У випадку першого входу на платформу користувач має зареєструватися, для чого продумано та створено дизайн сторінки реєстрації (рис. 3.7). Велика кількість полів вводу дозволить зібрати всі необхідні дані користувача. Пароль не має показуватися при введенні, а символи заміняються крапками, тож додано кнопку ока для деяких полів, що дозволить подивитися введений текст.

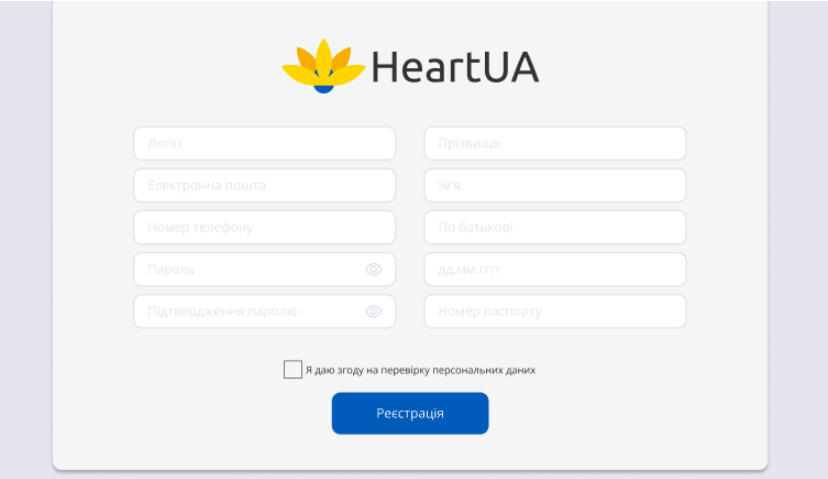


Рисунок 3.7 – Дизайн сторінки реєстрації

Після реєстрації користувач буде переведений на сторінку входу, де після введення логіну та паролю потрапить на головку сторінку. Перше що попадає у вічі користувачу є головний банер вебзастосунку та меню (рис. 3.8).

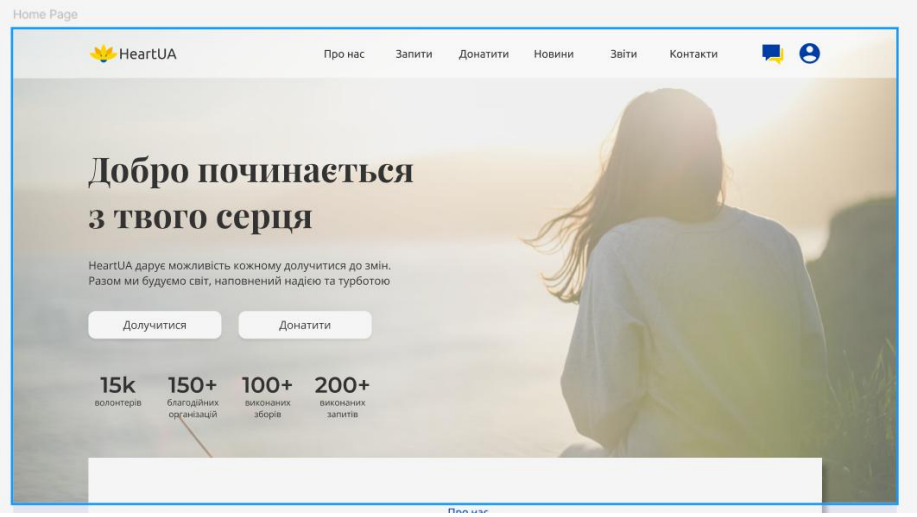


Рисунок 3.8 – Дизайн меню та головного банеру сайту

Головний банер відображає слоган проєкту, заклик приєднуватися, статистику платформи, до якої входить кількість волонтерів, кількість виконаних зборів, виконаних запитів, тощо. Кнопка долучитися перенаправляє користувача до сторінки реєстрації. Це необхідно у випадку, коли користувач зайшов на платформу

як гість. Фонове зображення банеру відображає глибину задуму, врівноваженість, доброту та спокій.

Меню складається з кнопок для переходу на інші сторінки та відкриття оверлеїв (елементів, що по суті накладаються поверх сторінку, як спливаючі вікна, та дозволяють застосувати різноманітні анімації).

До таких оверлеїв відноситься оверлей профілю (рис. 3.9), який складається з невеликої інформації про поточного користувача та посилання на підсторінки профілю, а також можливості переходу на сторінку контакти, або взагалі вийти з облікового запису.

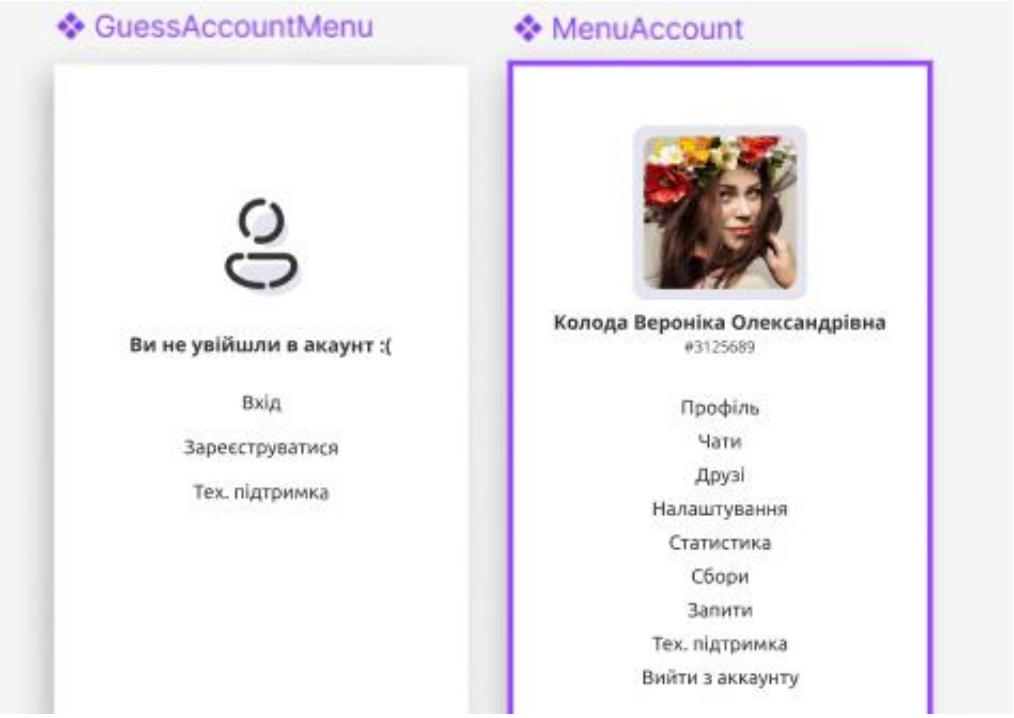


Рисунок 3.9 – Дизайн оверлею профілю у двох станах (при заходженні на платформу у ролі гостя та на свій обліковий запис)

Крім оверлею профілю, за допомогою меню можна відкрити оверлей чатів (рис. 3.10), на якому можна подивитися список чатів, користувачів, з якими ведеться спілкування та шматочок останнього повідомлення чату. При натисканні на сам чат можна відкрити повідомлення з конкретною людиною.

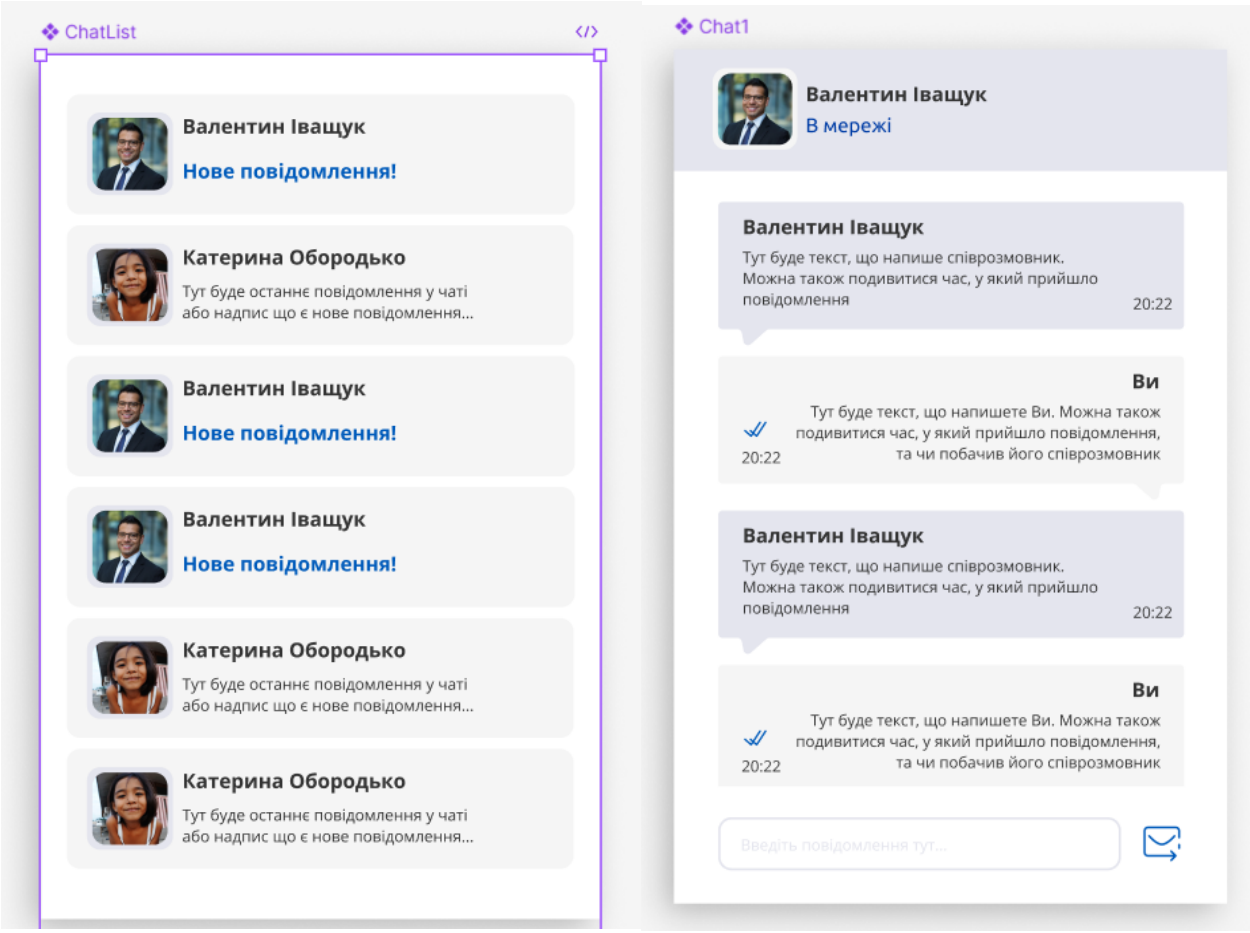


Рисунок 3.10 – Дизайн оверлею чатів

Для візуалізації відкриття оверлеїв застосовано анімацію певного типу «Open Overlay» з налаштуваннями швидкості прокрутки анімації, позиції відкриття оверлею та іншими.

Важливим блоком також є «Залишилося ще трішки…», який ілюструє можливість зробити грошовий вклад у благодійність на платформі, виділивши перші 3 збори, відсотковий показник яких вказує наскільки збір закритий (рис. 3.11). Користувач, що тільки знайомиться з платформою, зможе вже зробити свій вклад у благодійність, причому, можливо до кінця закривши збір, що стане сильним мотиватором для приєднання до спільноти волонтерів HeartUA.

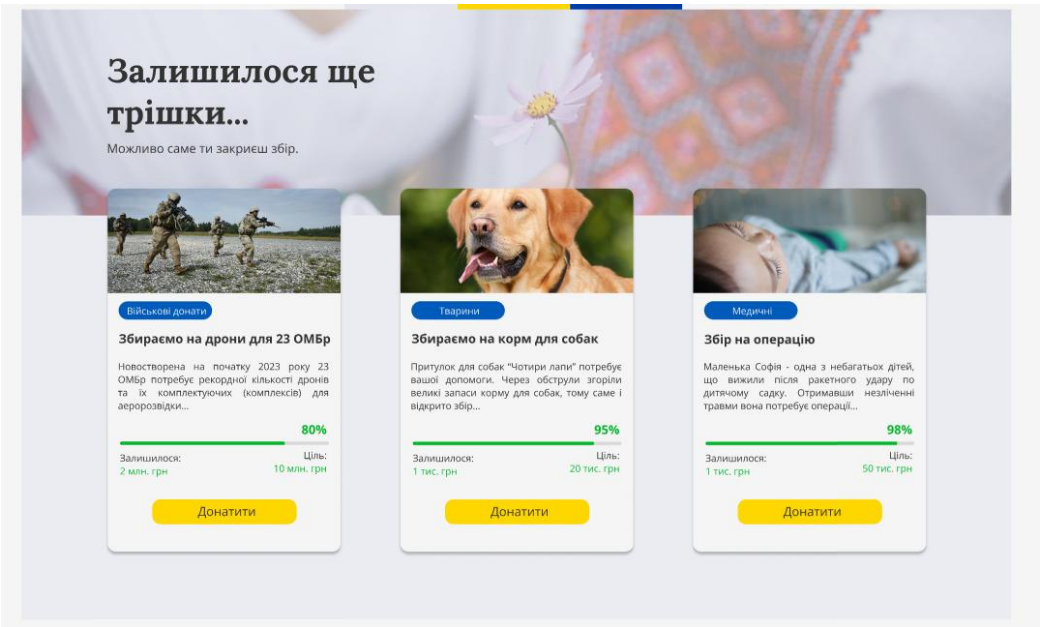


Рисунок 3.11 – Дизайн блоку «Залишилося ще трішки…»

Картка збору, представлена на рисунку 3.11, створена як компонент з декількома варіаціями (рис. 3.12). Вона містить скорочену інформацію про збір, тематику збору, ціль, процент закриття збору та кнопку, при переході на яку можна потрапити на сторінку конкретного збору.

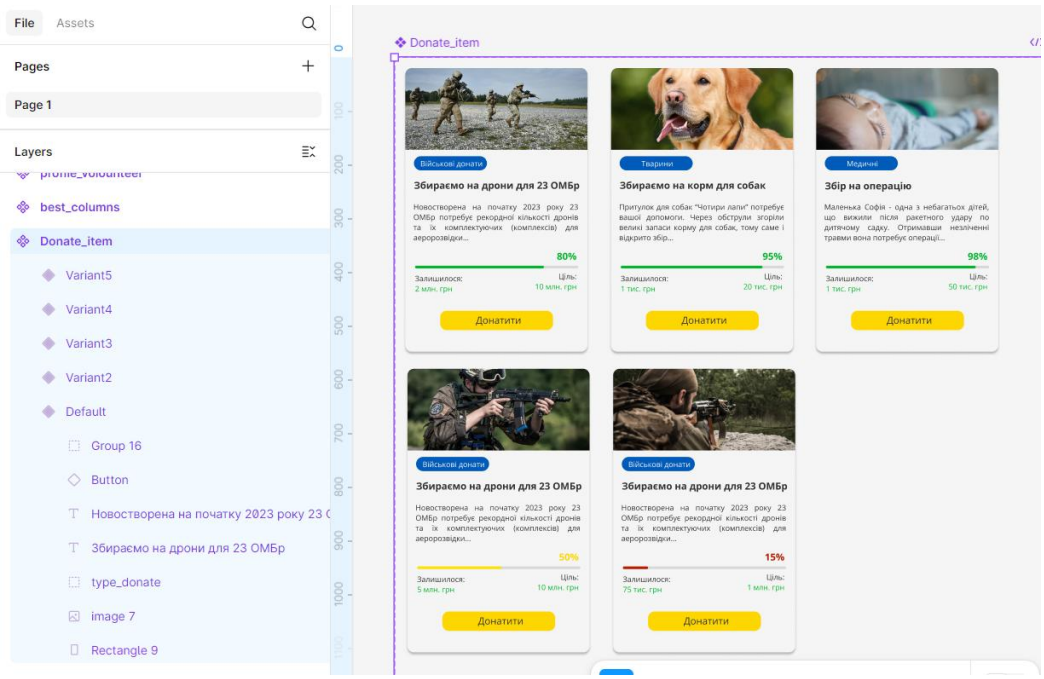


Рисунок 3.12 – Дизайн карток конкретного збору

Після блоку «Залишилося ще трішки…» йде мотиваційний блок, що закликає реєструватися на платформі та має кнопку переходу на сторінку реєстрації.

Ще одним блоком головної сторінки є блок «Спонсори» (рис. 3.13). У цьому блоці, крім списку спонсорів є можливість переказати кошти на розвиток проєкту, а також поділитися посиланням на платформу. Цей блок так само повторюється на сторінці «Про нас».

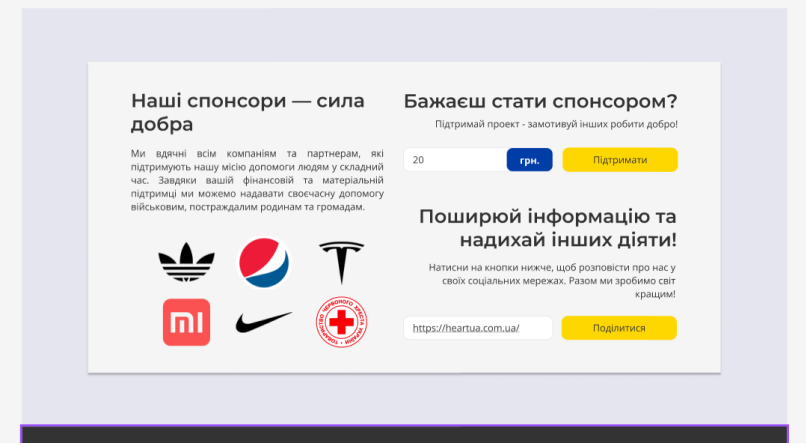


Рисунок 3.13 – Дизайн блоку «Спонсори»

Всі сторінки створено відповідно до структури вебзастосунку, подібно до створення головної сторінки. З компонентів, що виділяються варто вказати анімований логотип, створений за допомогою Figma, як динамічний компонент (рис. 3.14). Подібну анімацію застосовано так само до заголовків деяких сторінок.

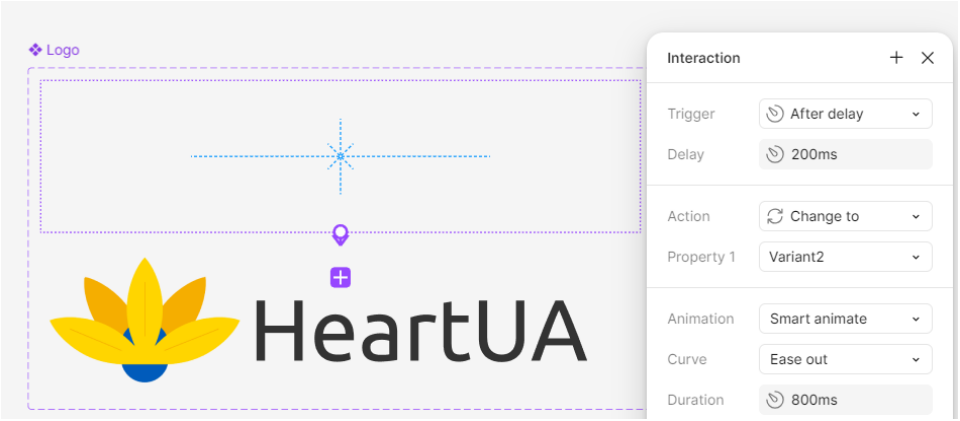


Рисунок 3.14 – Дизайн анімованого логотипу вебсторінки

Анімація карток розробників вебплатформи налаштовано таким чином, щоб при наведені з’являлася надпис з привітанням та цитатою розробника. Компонент складається з 2 частин: звичайного стану, та стану при наведенні (рис. 3.15). Також варто звернути увагу на використання анімації типу Smart animate, яка дозволяє використовувати положення елементу, який в даному випадку має з'явитися для забезпечення красивої та плавної анімації переміщення.

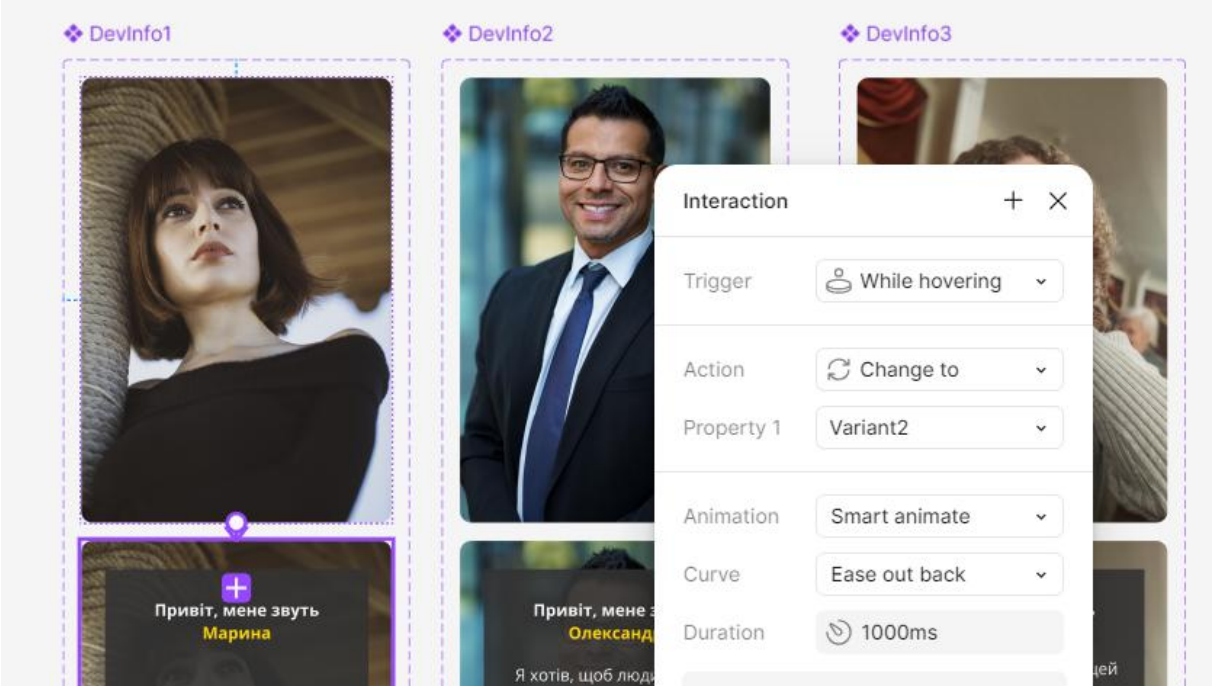


Рисунок 3.15 – Анімований елемент картка розробника

Дизайн проєкт містить велику кількість компонентів, слоїв, сторінок, анімації та переходів. Деякі компоненти неодноразово використані на різних сторінках. До таких належать кнопки, поля для вводу значень, картки, показники прогресу, меню, футер, тощо. Для демонстрації масштабу дизайн сторінки кожну сторінку розміщено у порядку рівнів, тобто спочатку головні сторінки, далі під ними дочірні, наприклад для сторінки зборів дочірньою є сторінка конкретного збору (рис. 3.16).

Рисунок 3.16 – Розміщення сторінок дизайн документу

Повністю ознайомитися з дизайн документом можна перейшовши за посиланням, вказаним у переліку джерел посилання [34].

Перехід між сторінками або компонентами також зображено у дизайн документі синіми стрілками. При натисканні на цю стрілку можна побачити характер переходу, анімацію та інші налаштування. У випадку поточного дизайн документу таких переходів доволі величезна кількість (рис. 3.17).

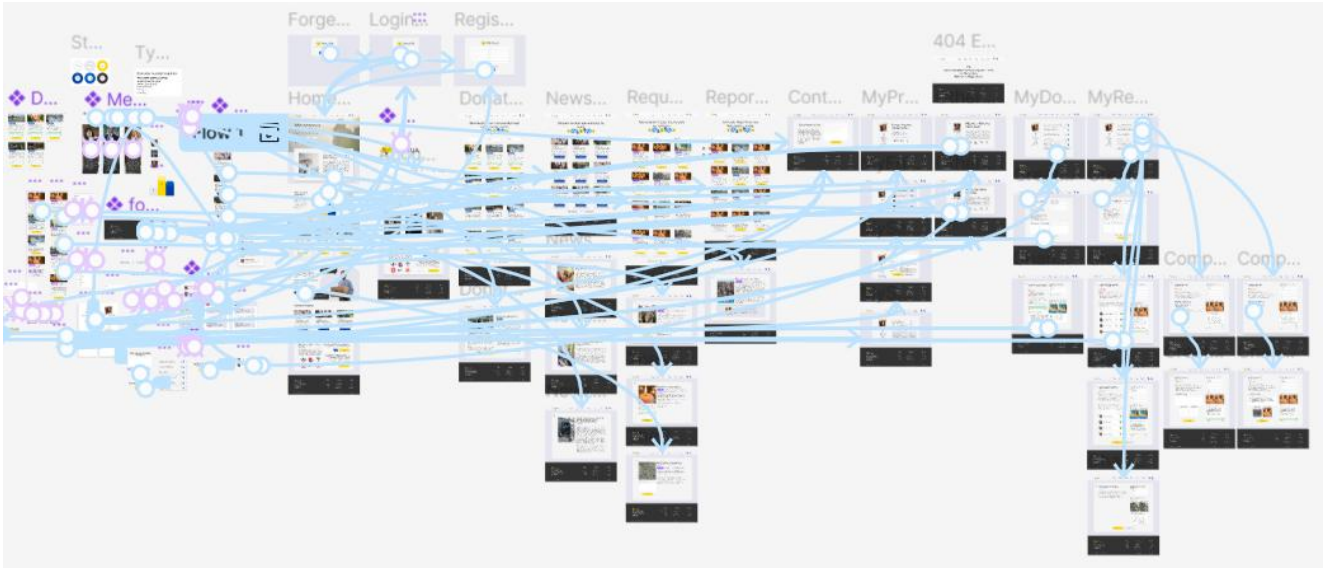


Рисунок 3.17 – Переходи між сторінками та компонентами у дизайн документі

Для зручності розуміння переходів між сторінками створено карту сайту за допомогою онлайн сервісу draw.io зазначену у додатку В.

Висновки до розділу 3

Розділ присвячено розробці інформаційної системи вебзастосунку.

Опис вхідних даних дав змогу проаналізувати необхідні сутності та дані, які активно використовуються на вебплатформі.

Опис структури системи допоміг структурувати різні функціональні можливості вебплатформи, визначитися з кількістю сторінок, необхідних для цілісної роботи системи, їх склад, тощо.

Етап проєктування був присвячений створенню БД з усіма необхідними таблиця та полями. Крім того, для забезпечення точної роботи системи у БД налаштовано зв’язки між таблицями, що дозволить контролювати збереження інформації, усунувши можливість додання до даних некоректних значень.

Етап моделювання складався з прийняття дизайнерських рішень, до яких входить вибір кольорів, шрифтів, назви вебзастосунку, з створення дизайн документу проєкту, який включає демонстрацію візуальної складової, наявність анімації та переходів. Додатково розроблено карту сайту, що безпосередньо буде у нагоді під час програмної розробки системи, адже дозволяє розбити проєкт на логічні взаємопов’язані частини. Крім того, створені під час створення дизайн документу, компоненти допомагають виділити елементи сайту, які неодноразово повторюються, що пришвидшить процес верстки вебплатформи.

В результаті створено дизайн документ проєкту, сформовано та налаштовано БД з усіма необхідними зв’язками.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

4.1 Опис програмної реалізації

Вебзастосунок «HeartUA» реалізовано за клієнт-серверною архітектурою, причому клієнтська частина написана з використанням фреймворку Vue.js та мовою програмування JavaScript, а серверну частину розроблено на PHP. Обмін даними між клієнтською та серверною частиною відбувається через REST API.

Для переходу між сторінками сайту застосовано клієнтський маршрутизатор (router), програмний код якого продемонстровано у додатку Г. Кожна сторінка є окремим компонентом, якому присвоєно певний маршрут, наприклад, щоб перейти на головну сторінку, треба перейти маршрутом /home. Такий спосіб реалізації переходів дає змогу показувати той чи інший компонент без перевантаження сторінок, причому покращуючи взаємодію між користувачем та вебплатформою.

Для того, щоб пришвидшити написання стилів для компонентів використано препроцесор SCSS, що дозволяє створити один базовий документ, де прописати стилі, що найчастіше використовуються у проєкт, такі як кольори, шрифти, стилі для кнопок, полів вводу, тощо. Повний код файлу розміщено у додатку Г.

Перша сторінка, яку бачить користувач, це сторінка авторизації (рис. 4.1).

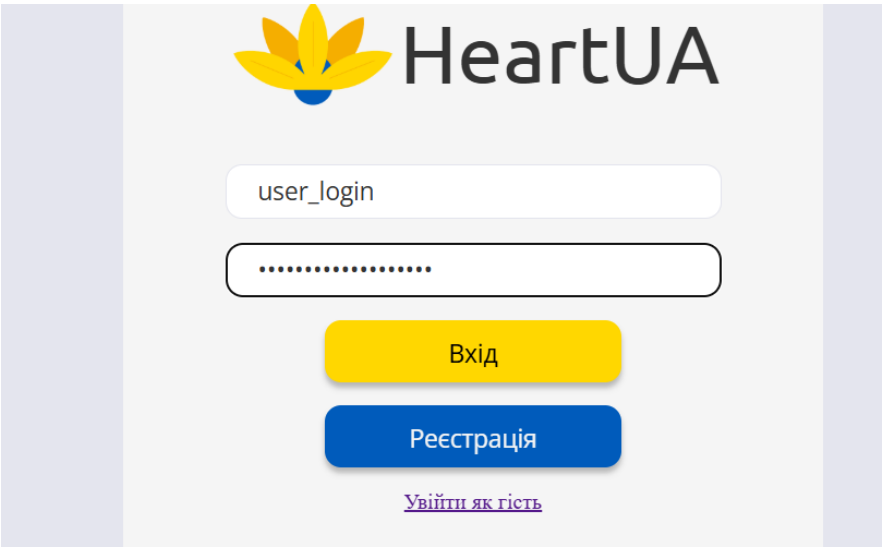


Рисунок 4.1 – Форма сторінки авторизації

Вона містить логотип, форму для входу, яка складається з полів вводу логіну та паролю, кнопки для авторизації, кнопки переходу на сторінку реєстрації та посилання, що дозволить перейти на платформу як гість. HTML розмітку створено з використанням стандартних тегів, описаних наступним фрагментом програмного коду (рис. 4.2).

```
<div class="sign-in-container">
  <div class="logo">
    <img src="../../assets/HeartUA_Logo.png" alt="Логотип" />
    <p>HeartUA</p>
  </div>
  <input v-model="login" type="text" placeholder="Login" />
  <input v-model="password" type="password" placeholder="Password" />

  <button class="continue" @click="openSession">Вхід</button>
  <button class="registration" @click="$router.push('/registration')">
    Реєстрація
  </button>
  <router-link to="/home">Увійти як гість</router-link>
</div>
```

Рисунок 4.2 – Структура сторінки авторизації

Для двостороннього зв’язку даних використано директиву v-model. Вона дозволяє оновлювати дані автоматично, у процесі того як користувач їх вводить, що дозволяє обійтися без додаткових обробок подій.

Для обробки події натискання на кнопку використано директиву @click, яка викликає певну функцію після натискання кнопкою миші.

Основна логіка сторінки описана у JavaScript документі та представлена у додатку Г. В цьому документі описується метод openSession(), який дозволяє передати на сервер введені користувачем логін та пароль для подальшої перевірки правильності введених даних.

Зв’язок з БД та подальші перевірки виконується PHP кодом, продемонстрованим у додатку Г, причому у разі неправильного введення тих чи інших даних сервер поверне повідомлення. Цей код виконує SQL запит, що

перевіряє наявність користувача у якого і логін і пароль буде співпадати з тим, що ввів користувач.

Якщо користувач на платформі вперше, то він перейде на сторінку реєстрації (рис. 4.3), яка, так само як і сторінка авторизації, складається з форми для введення даних.

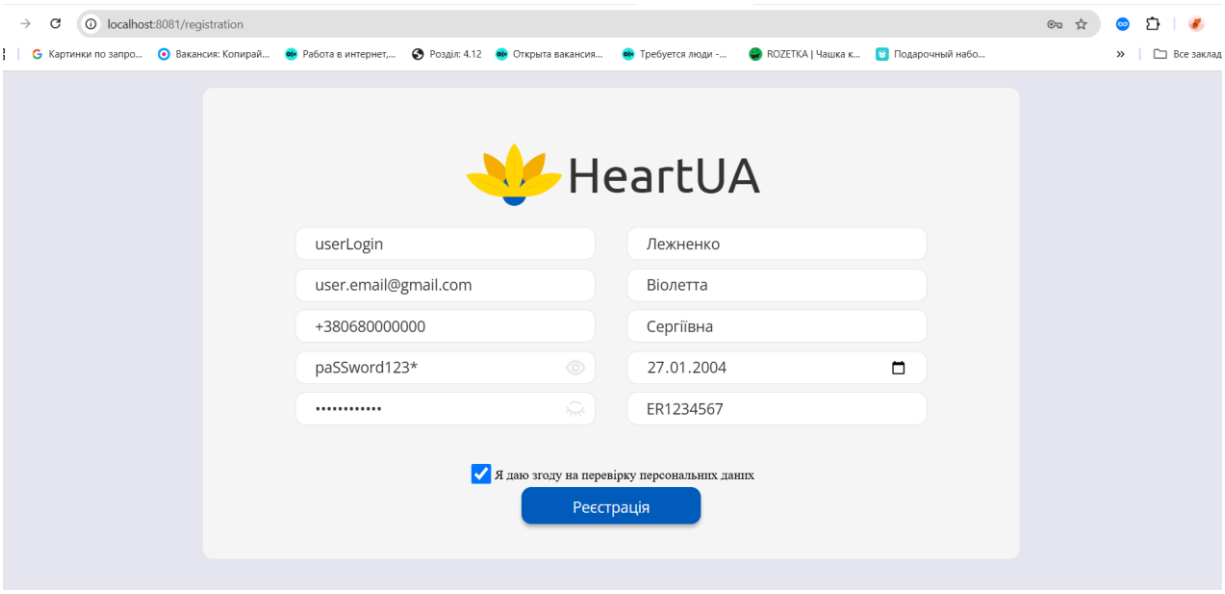


Рисунок 4.3 – Сторінка реєстрації нового користувача

JS документ, вказаний у додатку Г, описує перевірку введених користувачем даних за певними шаблонами, наприклад, для перевірки введеного номера телефона, електронної пошти, номера паспорту та паролю застосовано потрібні регулярні вирази. Крім перевірок правильності форматів даних перевіряється, чи існує такий логін та пошта у БД. За це відповідають функції isNewEmail() та isNewLogin(). Функція isUser18OrOlder() перевіряє, чи досяг користувач 18 років. Після перевірок, JS відправляє данні на сервер.

PHP файл сторінки реєстрації приймає данні, отримані з клієнтської частини вебзастосунку, підключається до БД, та, використавши SQL запит, додає у таблицю Users нового користувача.

Причому, варто зазначити, що для збереження номеру паспорту застосовано алгоритм шифрування AES-256 у режимі GCM, описаного у додатку Г. Функція encryptText(string $text, string $key) призначена для шифрування тексту (в даному випадку номер паспорту). Процес шифрування починається з того, що встановлюється шифр aes-256-gcm, після чого генерується випадковий ініціалізаційний вектор (IV), далі відбувається процес шифрування за допомогою openssl_encrypt, після чого буде отримано шифротекст та тег, до дозволяє перевірити цілісність даних при дешифруванні. Функція дешифрування decryptPassportNumber(array $data, string $key) розшифровує текст, використовуючи той самий ключ, що використано при шифруванні.

Компонент MainMenu відображає основне меню сайту (рис. 4.4). Воно зустрічається майже на кожній сторінці. Меню містить логотип, посилання для переходу між сторінками сайту та кнопки відкриття спливаючого вікна чатів та вікна меню профілю. Для переходу між сторінками застосовано метод $router.push(), яке відкривається при натисканні на кнопку для переходу.

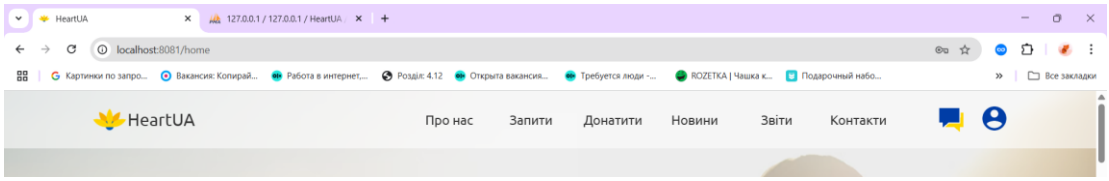


Рисунок 4.4 – Головне меню вебплатформи

Компонент MainFooter, що відображає нижню частину сайту, реалізовано подібним чином, тільки стилі задано по іншому (рис. 4.5). У складі компонента є велика кількість посилань, що дозволяють переходити на різні сторінки сайту.

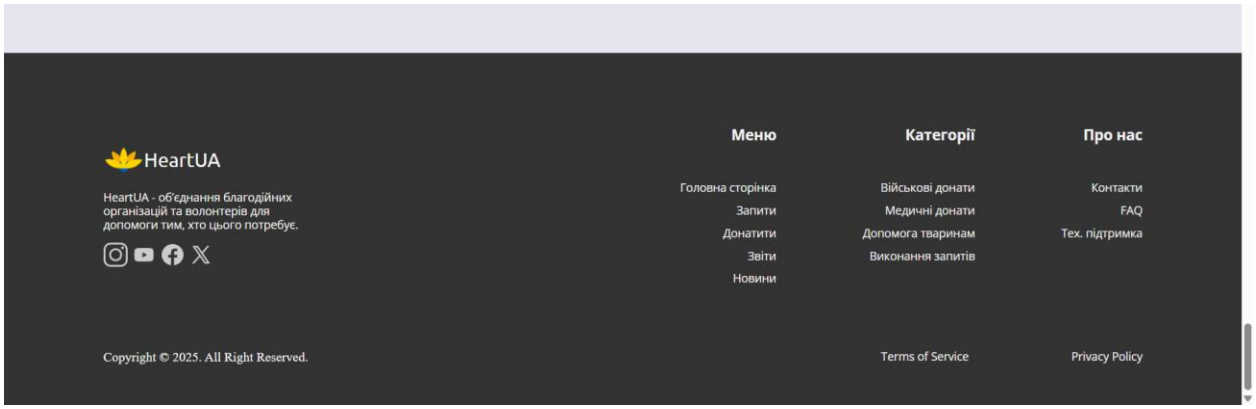


Рисунок 4.5 – Нижня частина сайту (футер)

Окрему увагу важливо приділити спливаючим вікнам, тобто чатам та меню профілю. Реалізації структури оверлею меню профілю (рис. 4.6) подібна до організації головного меню, але є особливості у стилях, заданих до даного компонента. Головною особливістю є застосування анімації. Для анімації використано CSS правило @keyframes, яке дозволяє гнучко налаштовувати анімацію, визначаючи її проміжні етапи. Так як оверлей виводить певну інформацію про користувача, що зайшов на платформу, тож компонент, під час першого відображення на сторінці відправляє на сервер ідентифікатор користувача та отримує з БД декілька необхідних для відображення даних.

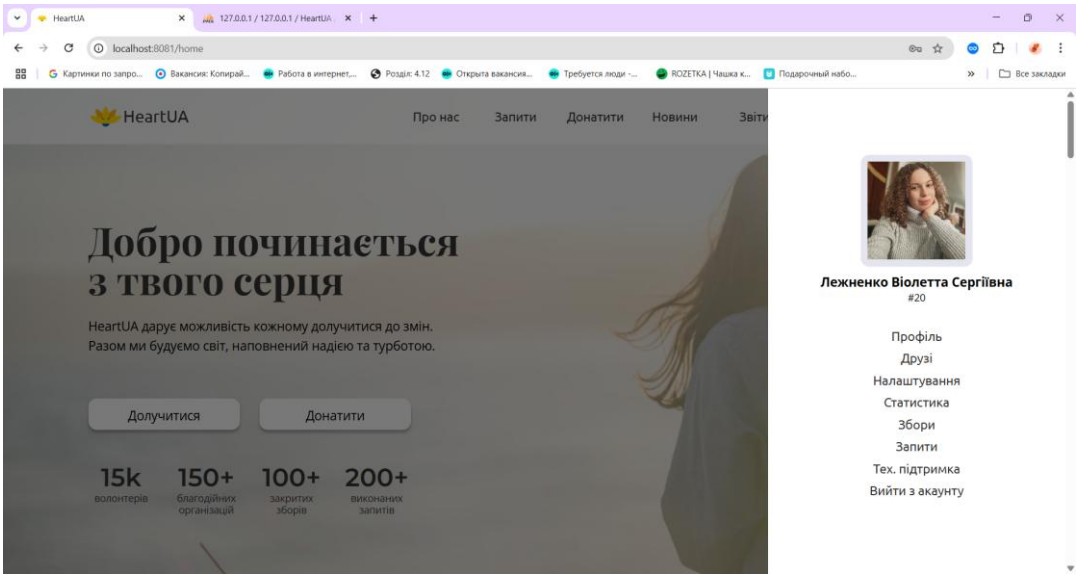


Рисунок 4.6 – Спливаюче вікно меню профілю

Спливаюче вікно чатів (рис. 4.7) користувача створено трішки іншим чином. Це вікно створено як компонент, що у свою чергу містить список компонентів TheChat, тобто список чатів. HTML документ, зазначений у додатку Г, як раз цей момент і описує.

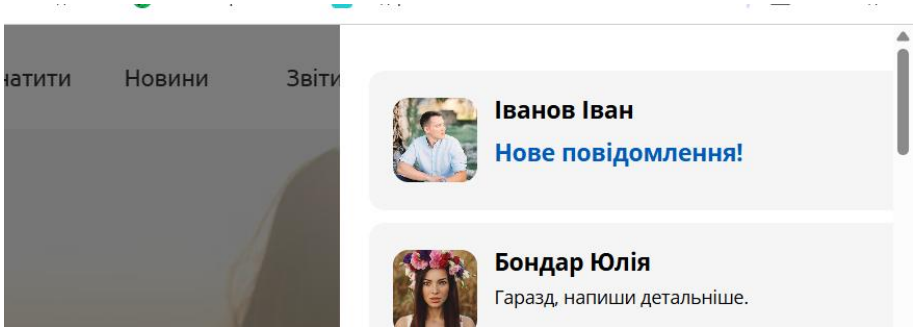


Рисунок 4.7 – Спливаюче вікно чатів

За допомогою JS коду передається на сервер ідентифікатор поточного користувача. Сервер робить запит до БД та повертає перелік чатів, а також, використовуючи дані з таблиці чатів у БД, отримано інформацію по кожному користувачу, з ким у поточного користувача є чат. SQL запит описано наступним програмним кодом (рис. 4.8).

```
$sql = "SELECT c.id,
IF(c.idUser1=:idU, c.idUser2, c.idUser1) as isPartner,
u.name,
u.lname,
u.sname,
u.image,
m.message,
m.status,
m.idSendUser
FROM Chats c
JOIN Users u ON u.id=IF(c.idUser1=:idU, c.idUser2, c.idUser1)
LEFT JOIN Messages m ON m.id=c.lastMessage
WHERE c.idUser1=:idU OR c.idUser2=:idU
ORDER BY
  CASE
    WHEN m.status = 'new' THEN 0
    WHEN m.status = 'read' THEN 1
    ELSE 2
  END;
";
```

Рисунок 4.8 – SQL запит отримання інформації про чати

Після отримання всіх необхідних даних, у структурі ми ці дані передаємо у дочірні компоненти TheChat використавши директиву v-for для того, щоб кожен з дочірніх компонентів отримав дані тільки про 1 чат. Крім того, важливо відмітити той факт, що, отримуючи статус останнього повідомлення з кожного чату («new» або «read»), змінюється і відображення компоненту.

Однією з важливіших деталей є також використання шини подій (Event Bus), тобто патерн, який дозволяє передавати якісь дані між незалежними компонентам, тобто не тільки між батьківськими та дочірніми. У випадку з спливаючим вікном чатів цей патерн використаний для оновлення чатів.

EventBus створить подію, що трапилася, у даному випадку передасть інформацію про те, що чати оновлено, а компонент ChatsOverlay, за допомогою певних методів, буде відслідковувати зміну цієї події і просто оновить список чатів. Це необхідно у разі наявності нових повідомлень. Коли користувач відкриє чат з новими повідомленнями ця подія спрацює і список чатів оновиться.

Після натискання на певний чат відкриються повідомлення з конкретним користувачем (рис. 4.9).

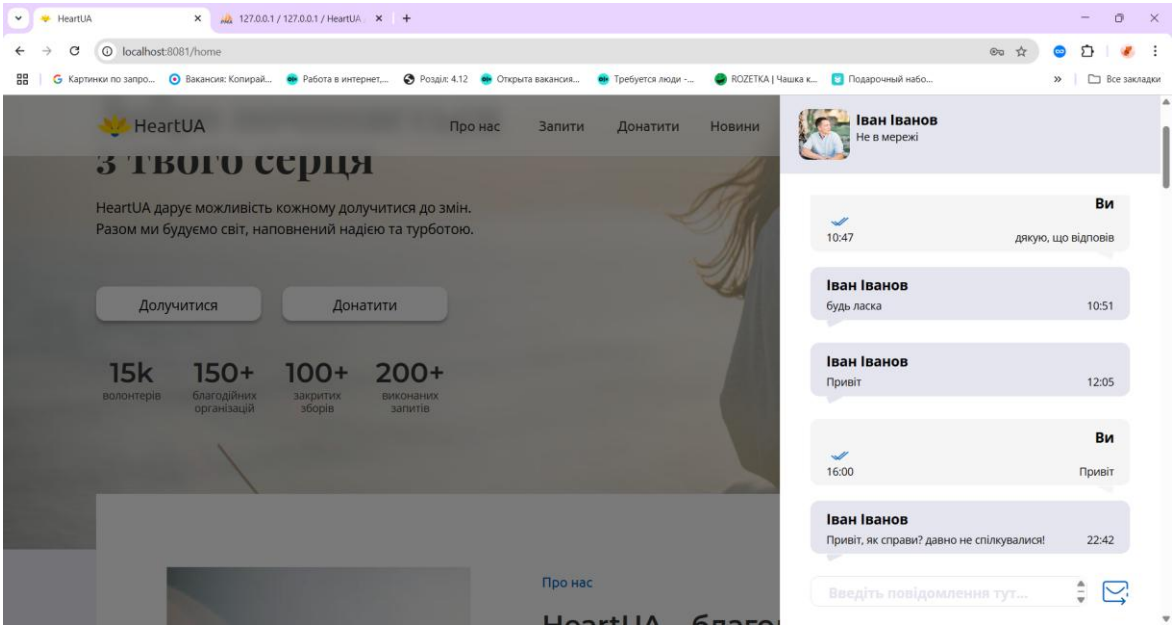


Рисунок 4.9 – Чат з конкретним користувачем

Так як повідомлень в одному чаті багато, тож структура розроблена подібним чином до списку чатів, тобто є один компонент MessageOverlay, тобто це наш чат, який у структурі показує за допомогою v-for почергово повідомлення, тобто компонент TheMessage. Крім цього передається у дочірній компонент всі необхідні дані для відображення. Додання дочірнього компоненту реалізовано наступним програмним кодом (рис. 4.10).

```
<div ref="scrollContainer" class="chat-list" @scroll="onScroll">
  <TheMessage
    v-for="m in messages"
    :key="m.id"
    :idUser1="m.idSendUser"
    :idUser2="m.idRecipient"
    :message="m.message"
    :dateTime="m.dateTime"
    :status="m.status"
    :name="name"
    :lname="lname"
  />
</div>
```

Рисунок 4.10 – Додання дочірнього компоненту

У JavaScript документі, програмний код якого вказано у додатку Г, описано основні функції, необхідні у чаті, такі як отримання даних про повідомлення з конкретним користувачем, відправку повідомлення та читання нових повідомлень.

Кожна з функцій відправляє необхідні дані на сервер для виконання тої чи іншої дії. Так наприклад для отримання інформації про всі повідомлення, відправлено на сервер ідентифікатор поточного користувача та співрозмовника, після чого на стороні сервера виконано SQL запит, описаний наступним програмним кодом (рис. 4.11).

```
$sql = "SELECT * FROM Messages
WHERE (idSendUser=:idU AND idRecipient=:idR)
OR (idSendUser=:idR AND idRecipient=:idU)";
$stmt = $dbcon->prepare(query: $sql);
$stmt->bindParam(param: ':idU', var: &$idU, type: PDO::PARAM_INT);
$stmt->bindParam(param: ':idR', var: &$idR, type: PDO::PARAM_INT);
$stmt->execute();
```

Рисунок 4.11 – Отримання повідомлень одного чату

Алгоритм відправки повідомлення відображено у якості PHP коду у додатку Г. Так як може бути ситуація, коли користувачі спілкуються вперше, тож крім додання конкретного повідомлення у таблицю Messages БД, треба й додавати новий чат між користувачами у таблицю Chats. У разі, якщо користувачі спілкується не вперше, то просто додається у БД нове повідомлення та оновлюється поле lastMessage таблиці Chats у БД.

Оновлення статусу нових повідомлень відбувається за рахунок SQL запиту, описаного наступним програмним кодом (рис. 4.12).

```
$stmt = $dbcon->prepare(query: "UPDATE Messages
SET status = 'read'
WHERE idSendUser = :idR
AND idRecipient = :idU
AND status = 'new'");
$stmt->execute(params: [':idR' => $idR, ':idU' => $idU]);
```

Рисунок 4.12 – Зміна статусу повідомлення

Причому запит буде оновлювати статус всіх повідомлень, які раніше були «новими».

Варто звернути увагу на використання PDO (вбудований клас у PHP, що дозволяє працювати з різними типами БД), що значно покращує гнучкість виконання запитів до БД використовуючи об’єктно орієнтований підхід.

Ще одним важливим функціоналом, реалізованим у цьому компоненті є механізм прокручування повідомлень. Якщо користувач заходить у чат, то він буде

бачити останні повідомлення. Крім того чат автоматично прокручується вниз при надсиланні повідомлення.

Головна сторінка платформи побудована з використанням примітивних блоків та кнопок переходів на інші сторінки сайту, хоча деякі з блоків сторінки потребують особливої уваги. До них відносяться блок «Залишилося ще трішки» (рис. 4.13), що відображає 3 збори, процентне відношення закриття яких є найвищими серед інших, та блок «Останні новини». Вони реалізовані подібним чином. За основу взято об’єкт карточку, що відображає найважливішу інформацію про збір, новину, тощо. Детальну реалізацію буде описано при розгляданні сторінок «Запити», «Донатити», «Новини» та «Звіти».

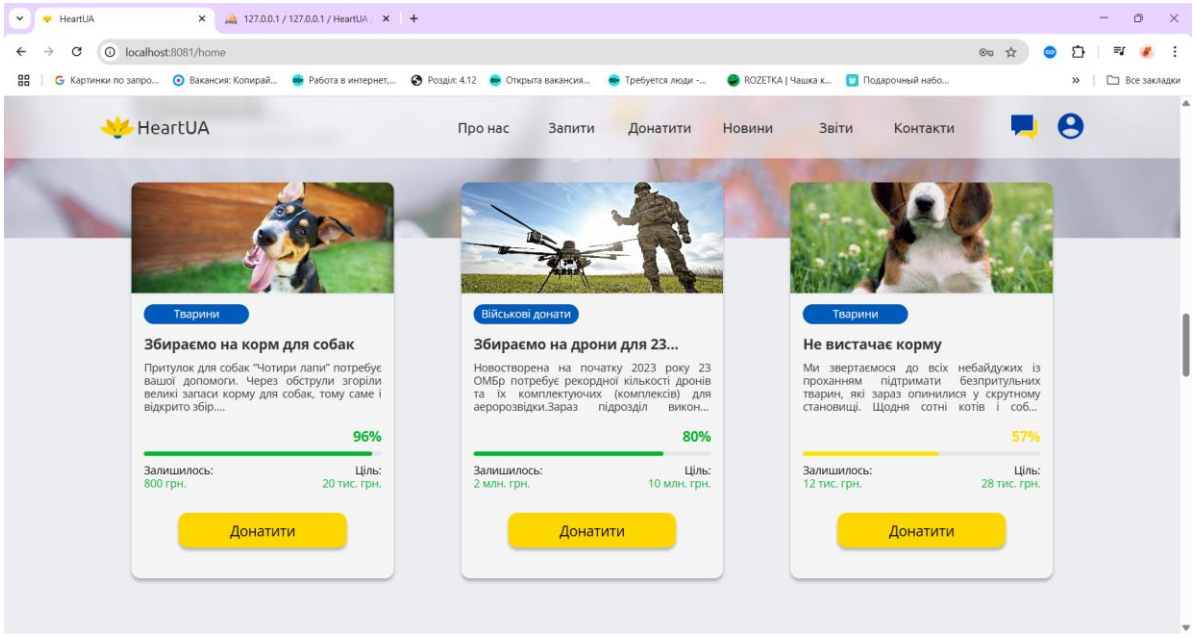


Рисунок 4.13 – Блок «Залишилося ще трішки»

Сторінка «Про нас» (рис. 4.14) містить інформацію про вебзастосунок та розробників. Картки розробників мають анімацію, що створена подібним чином до анімації спливаючих вікон чату та меню профілю, тобто застосовано CSS правило @keyframes, тільки застосовано більше стилів, таких як затемнення, поява та рух знизу вгору.

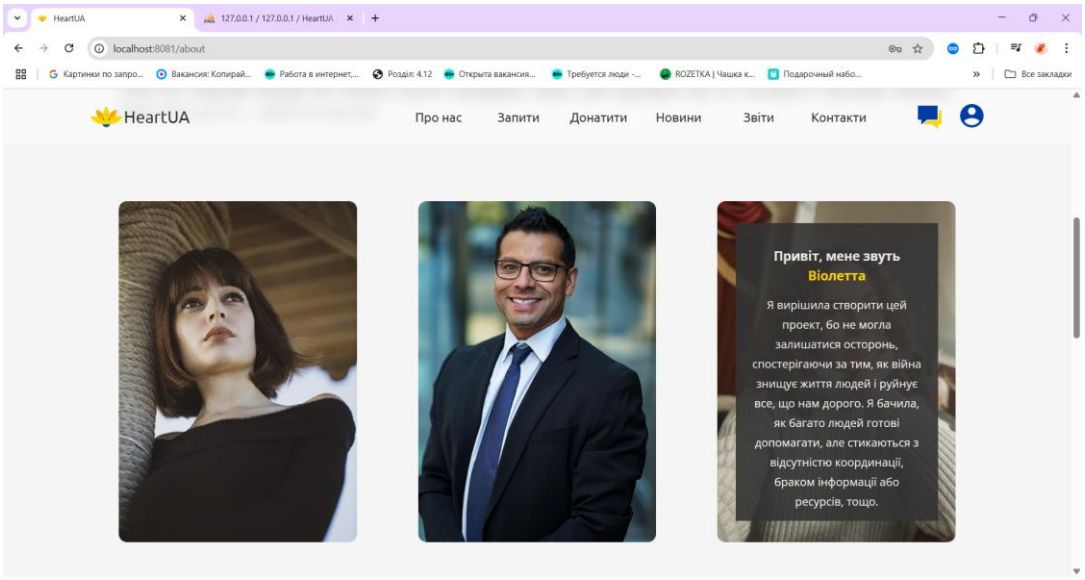


Рисунок 4.14 – Картки розробників вебсистеми

Сторінка зборів, основний програмний код якого продемонстровано у додатку Г, містить у собі перелік карток з короткою інформацією про збори. Структура сторінки побудована за принципом мотрійки, тобто компонент MonetaryFeesPage містить всі функціоналі елементи та компонент MonetaryFeesList, який у свою чергу робить запит на сервер, отримує список зборів та передає дані про кожен збір до компоненту MonetaryFeesCard, тобто MonetaryFeesList складається з багатьох компонентів MonetaryFeesCard (рис. 4.15).

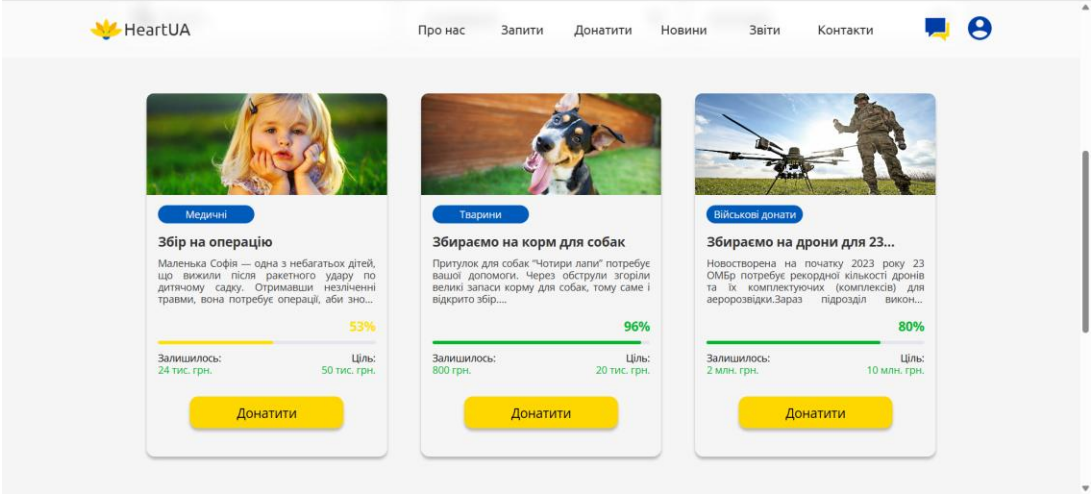


Рисунок 4.15 – Картки сторінки зборів

MonetaryFeesCard, отримавши інформацію, трішки модернізує вивід даних про ціль збору та залишок, використавши функції, описані таким програмним кодом (рис. 4.16).

```
computed: {
  formattedGoal() {
    if (this.goal > 1000 && this.goal < 1000000) {
      return (this.goal / 1000).toFixed(0) + " тис.";
    } else if (this.goal > 999999) {
      return (this.goal / 1000000).toFixed(0) + " млн.";
    }
    return this.goal;
  },
  formattedRest() {
    if (this.rest > 1000 && this.rest < 1000000) {
      return (this.rest / 1000).toFixed(0) + " тис.";
    } else if (this.rest > 999999) {
      return (this.rest / 1000000).toFixed(0) + " млн.";
    }
    return this.rest;
  },
},
```

Рисунок 4.16 – Модернізація виводу залишку та цілі

Ці функції необхідні для того, щоб в одну картку можна було вмістити відображення великих сум коштів, наприклад тисячі та мільйони.

Повернувшись до компоненту MonetaryFeesList, варто зазначити, що у ньому прописана і вся реалізація пагінації сторінки. Тобто на сайті показується певна кількість карток зборів, а після натискання кнопки «Показати ще», з сервера підгружається ще стільки ж.

Поточна пагінація працює таким чином. На сервер відправляється кільки карток, яку треба дозавантажити на сайт, та номер групи. На PHP ми створюємо запит, у якому вказуємо ці значення. В результати всі запити з таблиці у БД розбиваються на групи по n кількість, тобто таку, яку треба показати за одну підгрузку на сайт. За номером групи знаходиться які саме картки треба підгрузити і сервер відправляє їх. В результаті клієнтська частина оновлюється і можна побавити більше карток зі зборами.

Окремо варто приділити увагу пошуку та сортуванню карток. Пошук карток можна здійснити за id та за назвою. Сортувати можна за ціллю або новизною збору (рис. 4.17), чи категорією (рис. 4.18).

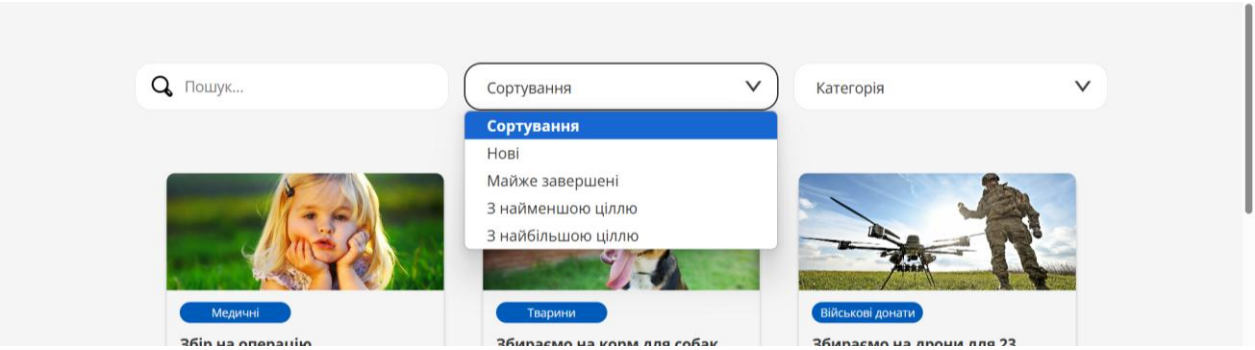


Рисунок 4.17 – Сортування за ціллю збору та новизною

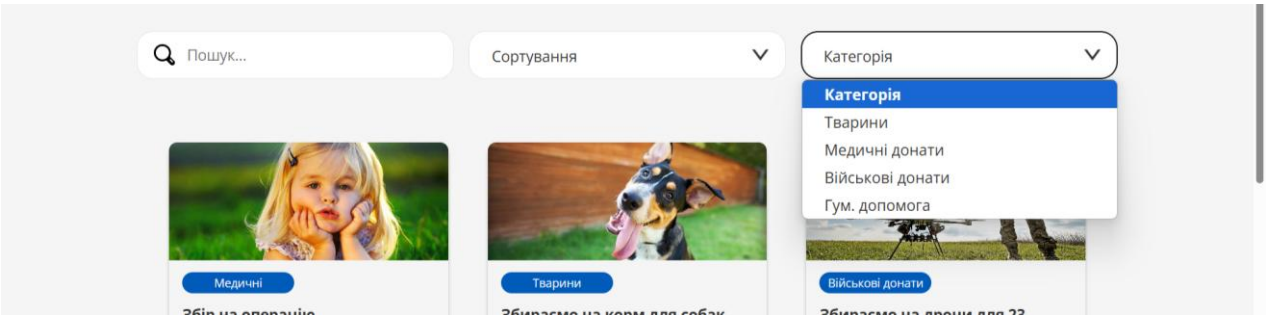


Рисунок 4.18 – Сортування за категорією

Кожен з вище зазначених фільтрів відслідковується за допомогою реактивного спостерігача (watcher), або по іншому можна назвати як опція watch. Вона відслідковує зміни у цих компонентах (у випадку з пошуком це зміна тексту, а у випадку з випадаючими списками це вибір іншого елементу) та перезавантажує список зборів, використавши відправку на сервер даних фільтрів.

Також потребує уваги застосування mapping, реалізованому у JS коді компоненту MonetaryFeesList. Mapping оброблює отримані дані, в даному випадку, цей підхід дозволив обрахувати процент закриття збору та кількість грошей, що ще треба зібрати.

При натискання на кнопку «Донатити» здійснюється перехід на сторінку конкретного збору (рис. 4.19).

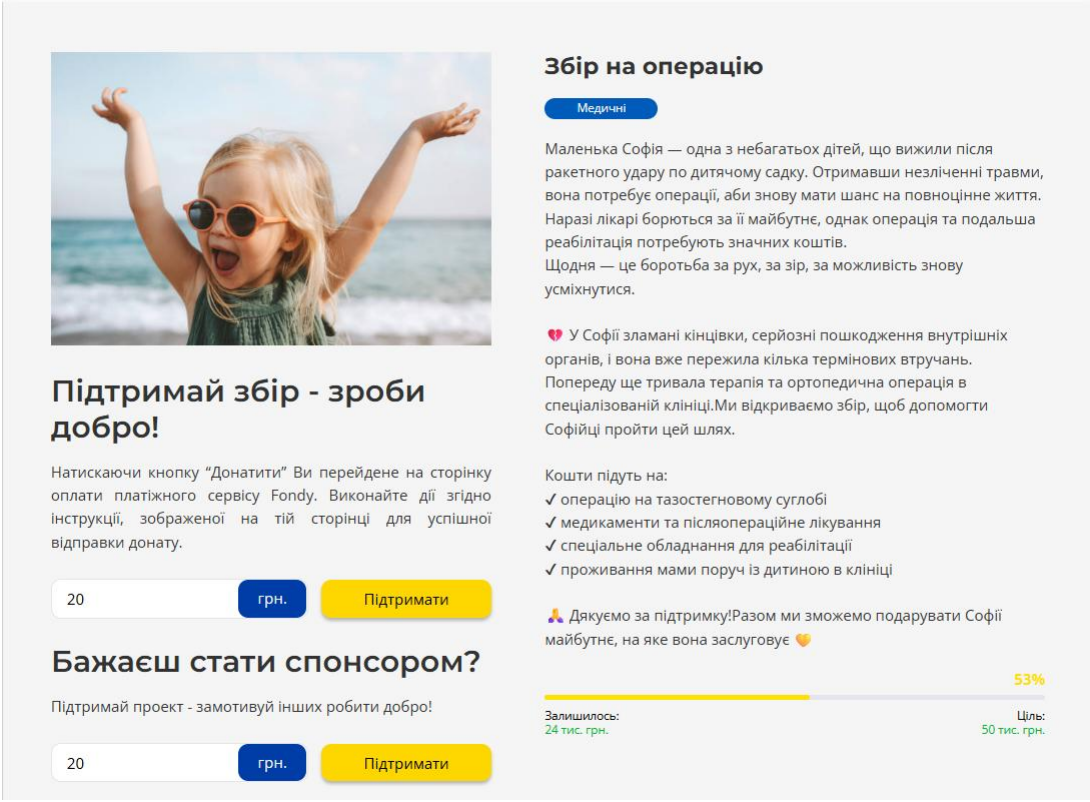


Рисунок 4.19 – Сторінка конкретного збору

Найцікавіша частина даної сторінки це 2 поля вводу та кнопки, що необхідні для переходу до платіжної системи Fondy та здійснення платежу. Цей процес описаний у функціях goToPaymentSystem() та goToPaymentSystemSponsors(), для того щоб перевести гроші на збір та на проєкт. Друга функція неодноразово використовується на сайті, зокрема на головній сторінці у блоці «Спонсори» та на сторінці «Про нас».

Ці функції передають на сервер необхідні дані. Сервер ці дані отримує і на основі них заповнюються всі необхідні поля платіжного запиту на Fondy. До таких полів належать:

– merchant_id – ID продавця, у даному випадку не обов’язкове поле;
– order_id – унікальний ID замовника;

– amount – сума коштів на переведення (у копійках);
– currency – валюта;
– order_desc – опис замовлення;
– response_url – посилання, куди перейду користувач після успішної оплати;
– signature – підпис (необхідний для безпеки та перевірки цілісності даних).

Після цього запит відправляється на Fondy, і якщо все заповнено вірно, то Fondy поверне посилання на сторінку здійснення платежу (рис. 4.20).

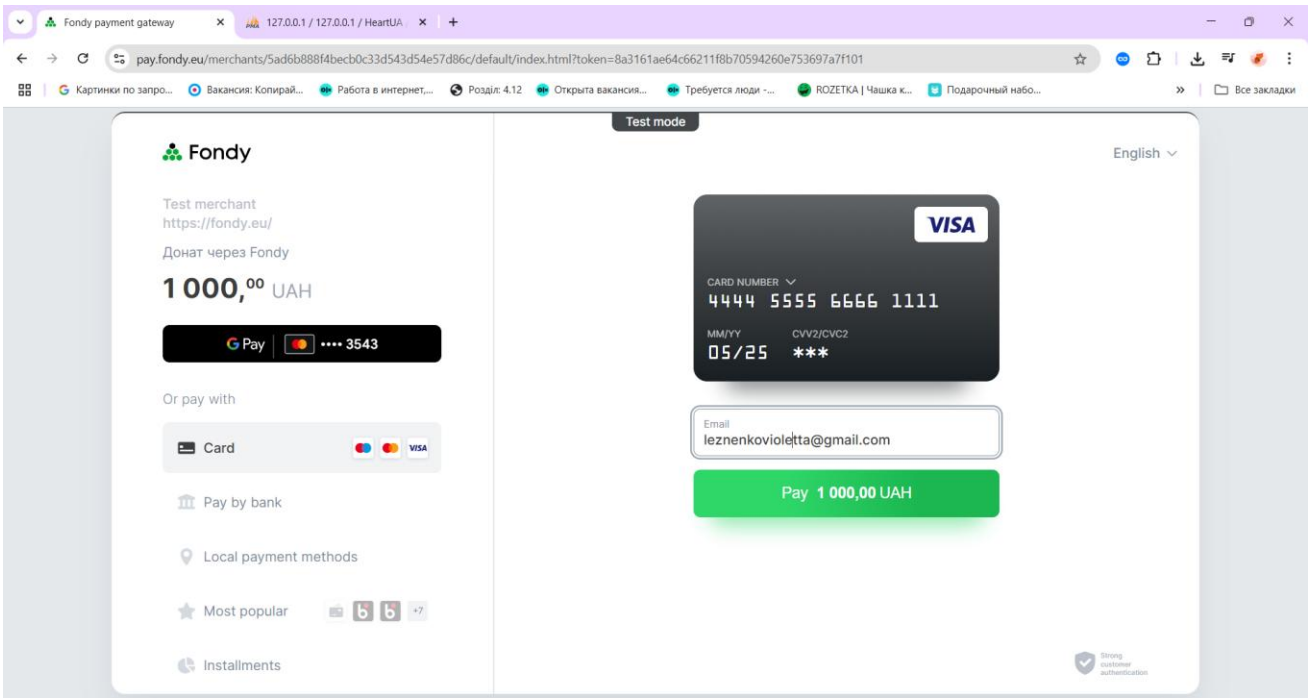


Рисунок 4.20 – Сторінка для здійснення платежу

Після успішно проведеної оплати користувач бачить наступну сторінку (рис. 4.21), хоча на стороні сервера було здійснено ще декілька процесів, до яких входить оновлення суми, зібраної на певний збір, перевірка закриття збору і у разі, якщо збір закрито, то створюється пустий шаблон для звіту. Такий шаблон необхідний для того, щоб користувач, що створив збір зміг звернути увагу.

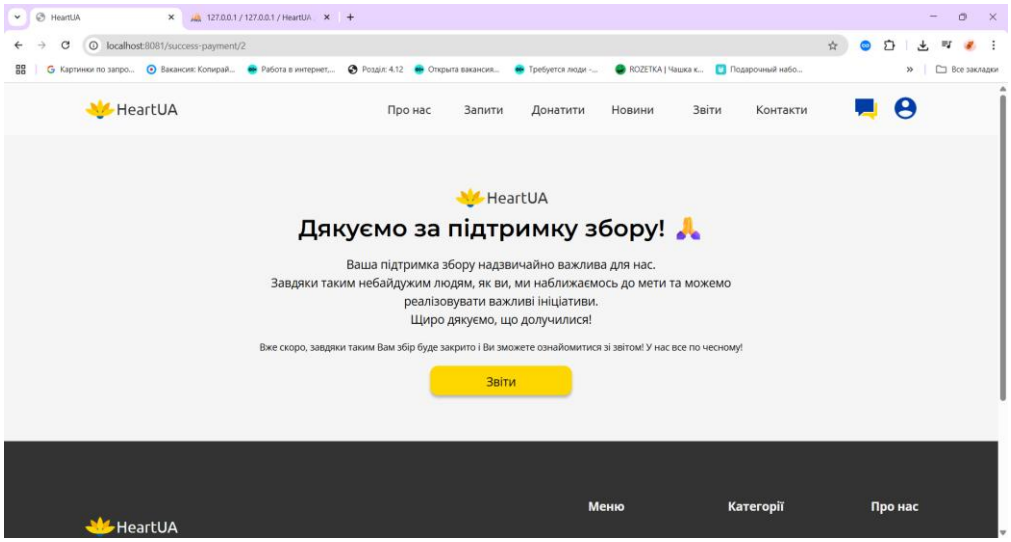


Рисунок 4.21 – Сторінка подяки за донат на певний збір

У разі, якщо користувач перевів кошти на сам проєкт, то в результаті він побачить подібну сторінку, трішки з іншим текстом, але суть залишається та сама.

Ще одним приємним бонусом від фонді було те, що при відправки донату на збір можна запросити надсилання квитанції про оплату на пошту, де видно усю інформацію по здійсненому платежу (рис. 4.22).

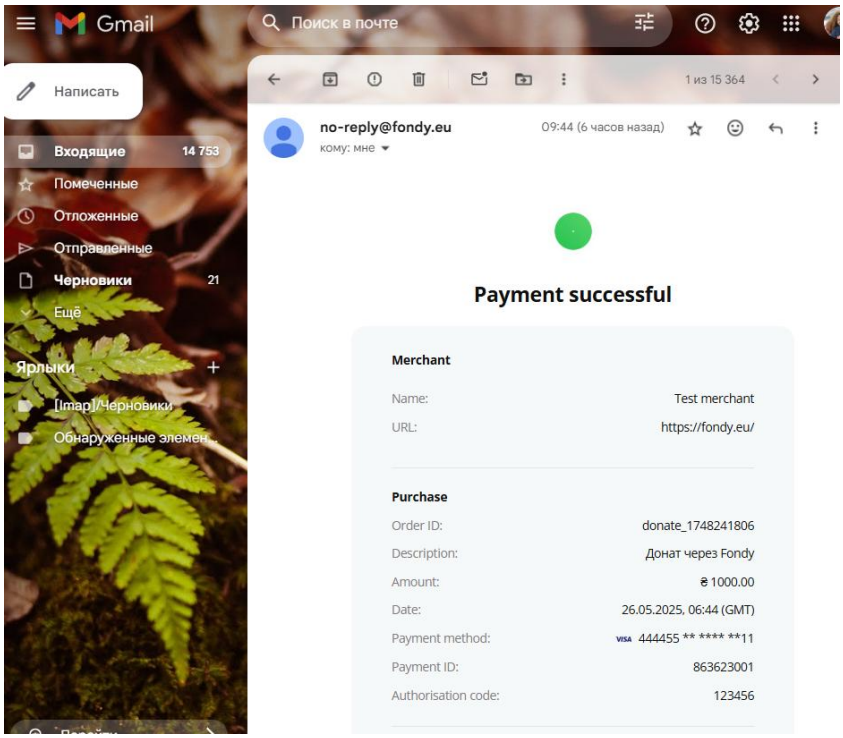


Рисунок 4.22 – Лист-квитанція про здійснений донат

Сторінка «Запити» реалізовано подібно до сторінки зборів (рис. 4.23). Використано той самий принцип мотрійки. Так само можна застосувати фільтрацію за типом запиту, локацією, пошук за id та назвою. Також реалізовано пагінацію. Єдине що відрізняються стилі карток, так як на цей час є вибір серед 3 типів запитів, а саме «Матеріали», «Фіз. Допомога» та «Порада», з яких «Порада», не має цілі, тому й відсоток і остача для такого запиту не розраховується.

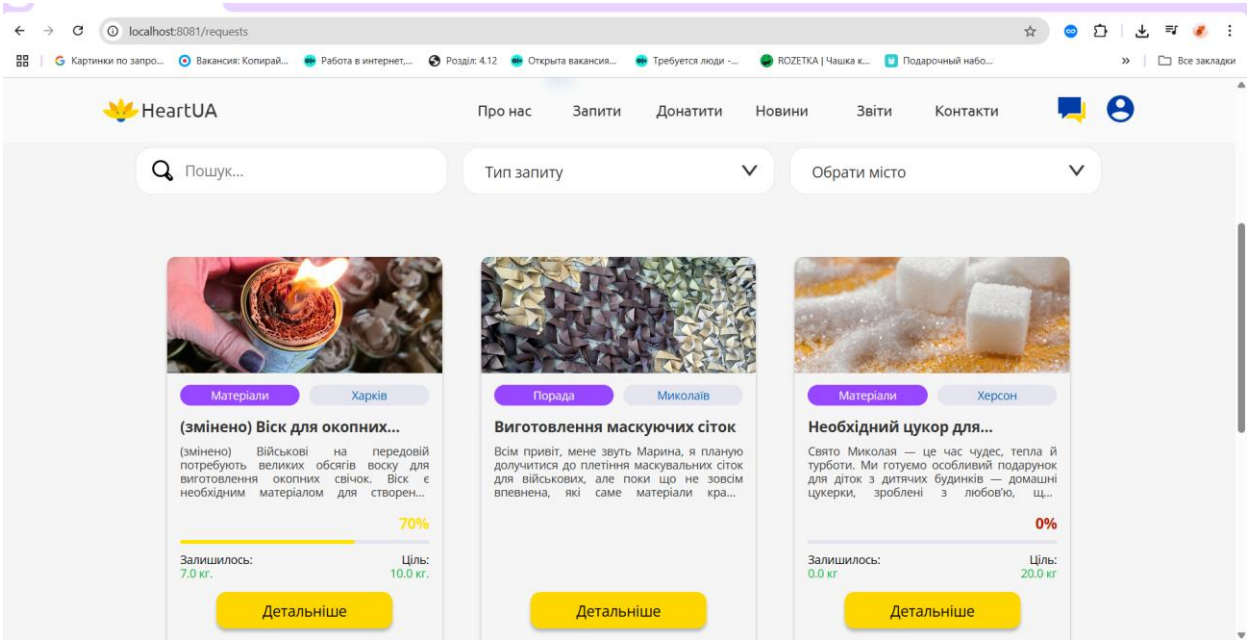


Рисунок 4.23 – Список запитів на сторінці «Запити»

Сторінка окремого запиту (рис. 4.24) схожа структурою на сторінку окремого збору, але має одну відмінність. Якщо взяти тип запиту «Матеріали», то на сторінці з’явиться поле для вводу кількості матеріалу, якою користувач захоче поділитися, та відповідна кнопка. У випадку з запитом «Фіз. допомога», то буде видно функціональну кнопку «Записатися», а ось якщо запит типу «Порада», то буде видно поле для вводу багаторядкового тексту та кнопка відправити пораду.

JS логіка сторінки описує 3 функції відправки пропозиції допомоги в залежності від запиту. Загалом ці три функції однаково збираються до сервера, єдине що з різним набором отриманих даних для додавання їх у БД.

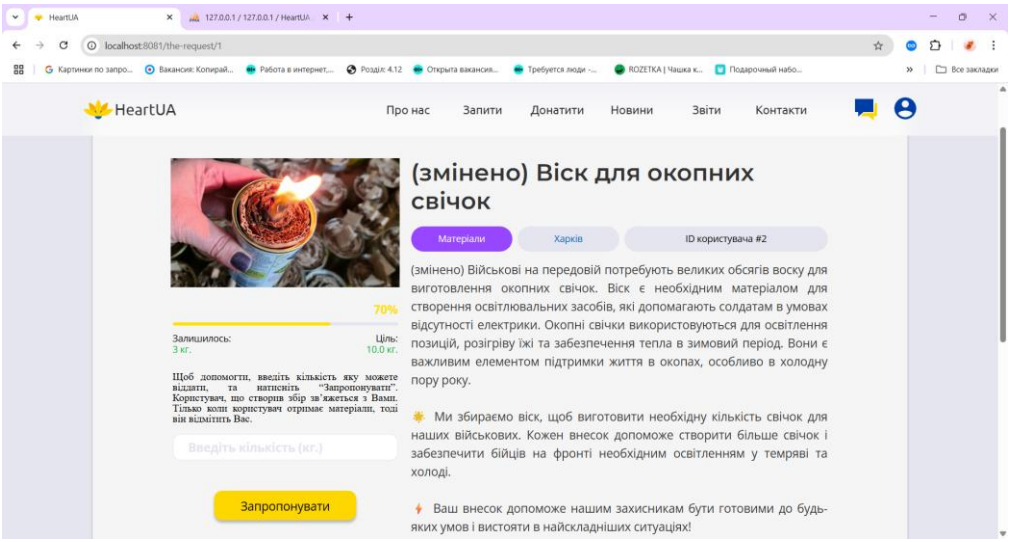


Рисунок 4.24 – Сторінка окремого запиту

Сторінка «Новини» та сторінка зі звітами реалізовано за схожим принципом, як і сторінки з запитами та зборами.

Сторінка «Контакти» необхідна для зв'язку з користувачами (рис. 4.25). Вона мітить форму, частина якої автоматично заповнюється даними, такими як ім'я та електронна адреса, у випадку, якщо користувач зареєстрований на платформі.

JS логіка даної сторінки описує функцію getUserInfo(), що дозволяє отримати дані зареєстрованого користувача, та функцію sendEmail(), яка відправляє дані на сервер для подальшої відправки повідомлення.

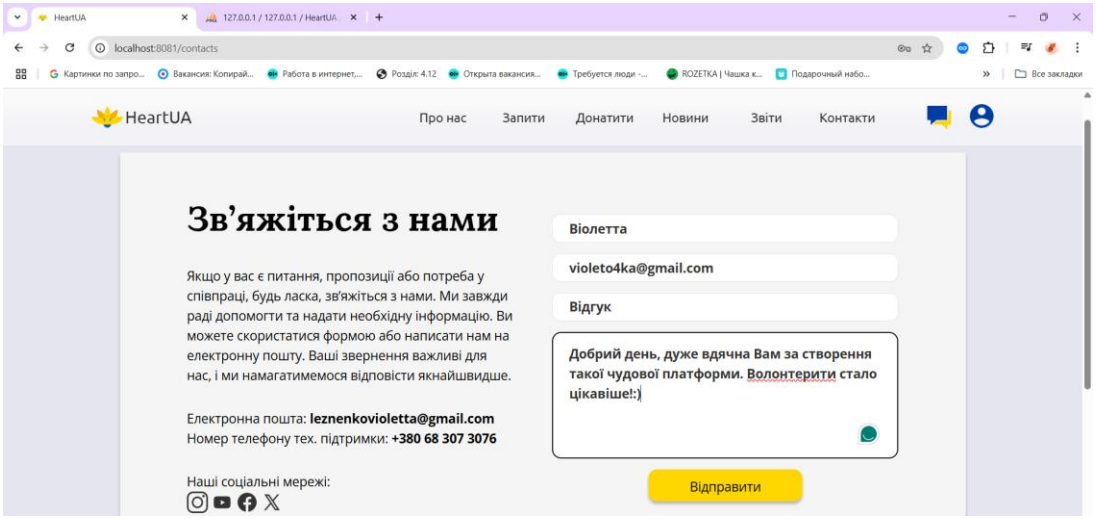


Рисунок 4.25 – Сторінка «Контакти»

Код на PHP, представлений у додатку Г, виконує безпосередню відправку повідомлення. Для роботи з поштовим сервісом ukr.net використано бібліотеку PHPMailer. Для листа обрано варіант HTML, що дозволяє надати електронному листу певного стилю. В результаті заповнення полів, зображених на рисунку 4.25, та відправки цих даних, на пошту прийшло наступне повідомлення (рис. 4.26).

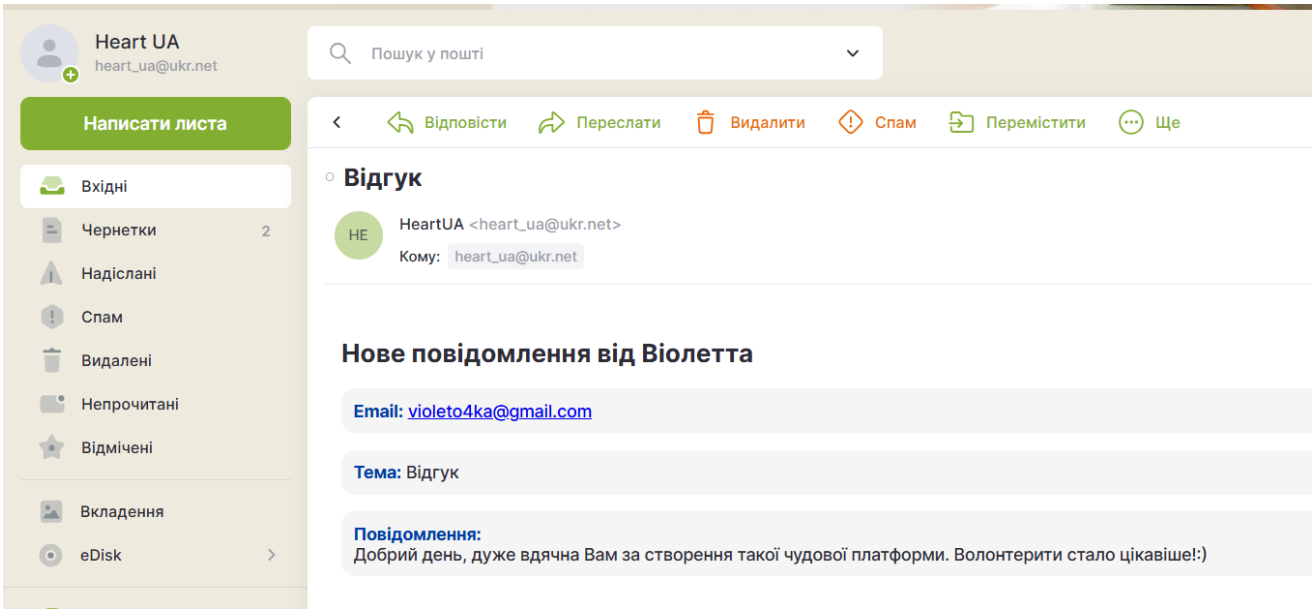


Рисунок 4.26 – Відправлений лист з використанням поштового сервісу ukr.net

До сторінки іншого користувача (рис. 4.27) створена доволі мінімалістичною. На сторінці є інформація про іншого користувача, отримана з серверу по id, та 3 кнопки: «Написати», «Додати у друзі» або «Видалити з друзів», та «Поскаржитися». Саме тому у JS документі описано наступні функції:

– showChat() – відкриває спливаюче вікно чату з іншим користувачем;

– fetchUserInfo() – для отримання інформації про іншого користувача;

– checkFriend() – перевірка, чи дружить поточний користувач з іншим користувачем;

– deleteFriend() – видаляє того іншого користувача з друзів;

– addFriend() – додає користувача у друзі.

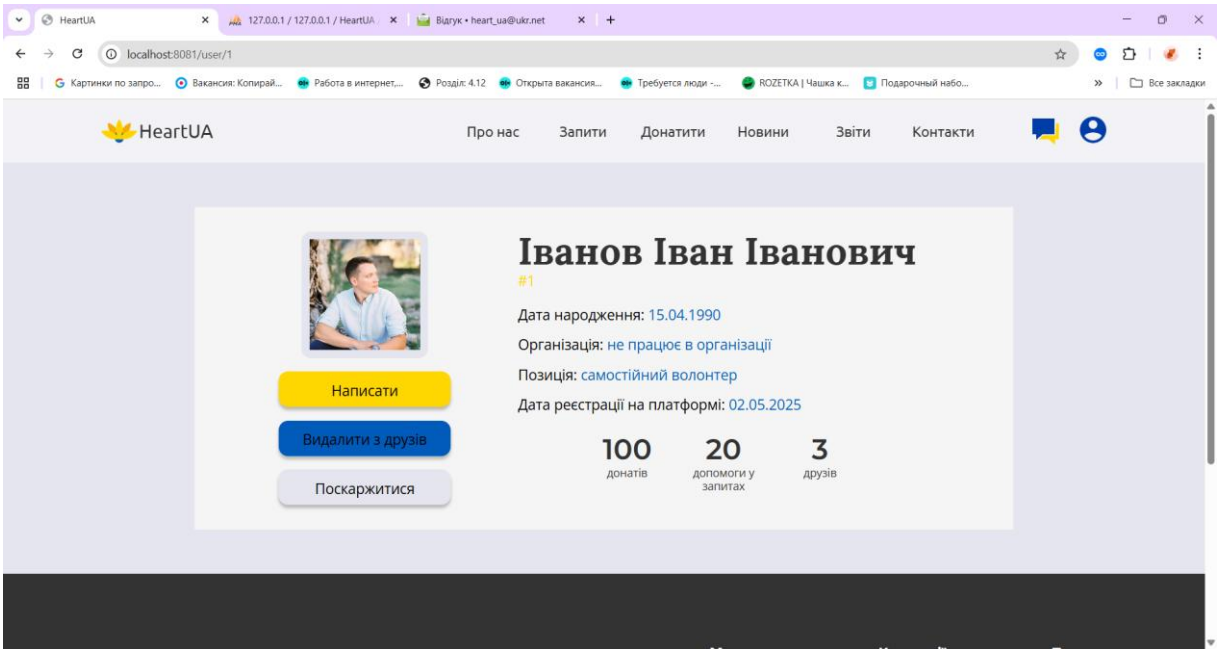


Рисунок 4.27 – Сторінка іншого користувача

Сторінка профілю містить декілька підсторінок, тому є основна сторінка, яка містить меню та тег <router-view />, на місце якого додається той чи інший компонент, до якого переходить користувач. Перша з цих підсторінок с сторінка з інформацією про поточного користувача (рис. 4.28).

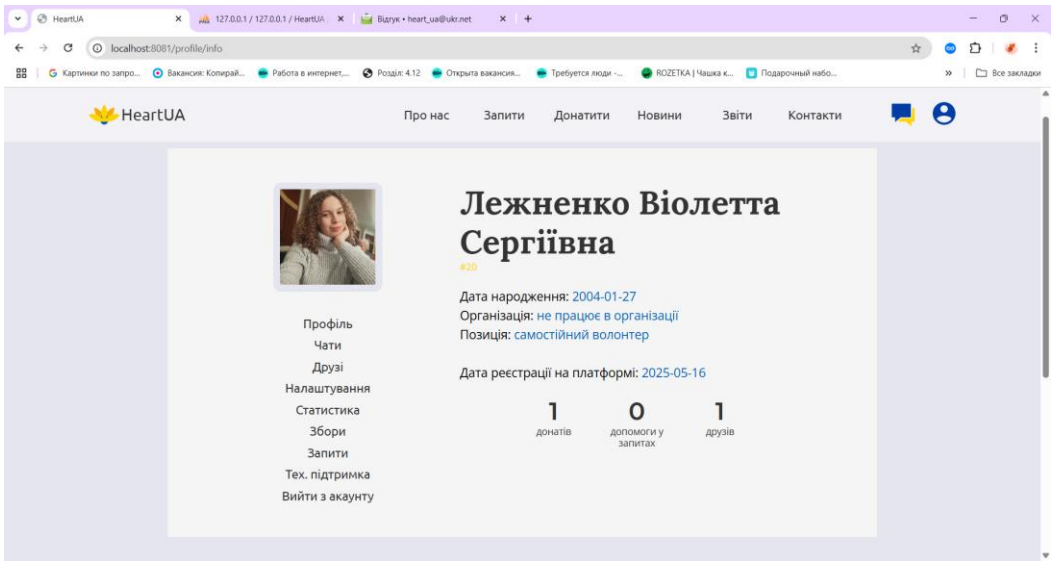


Рисунок 4.28 – Сторінка профілю про поточного користувача

Якщо натиснути на Чати то відкривається раніше згадане спливаюче вікно чатів.

Варто згадати про підсторінку «Налаштування» (рис. 4.29). Тут є декілька полів вводу, причому для деяких встановлено властивість readonly, тобто тільки для читання, це ті поля, інформацію у яких змінювати не можна.

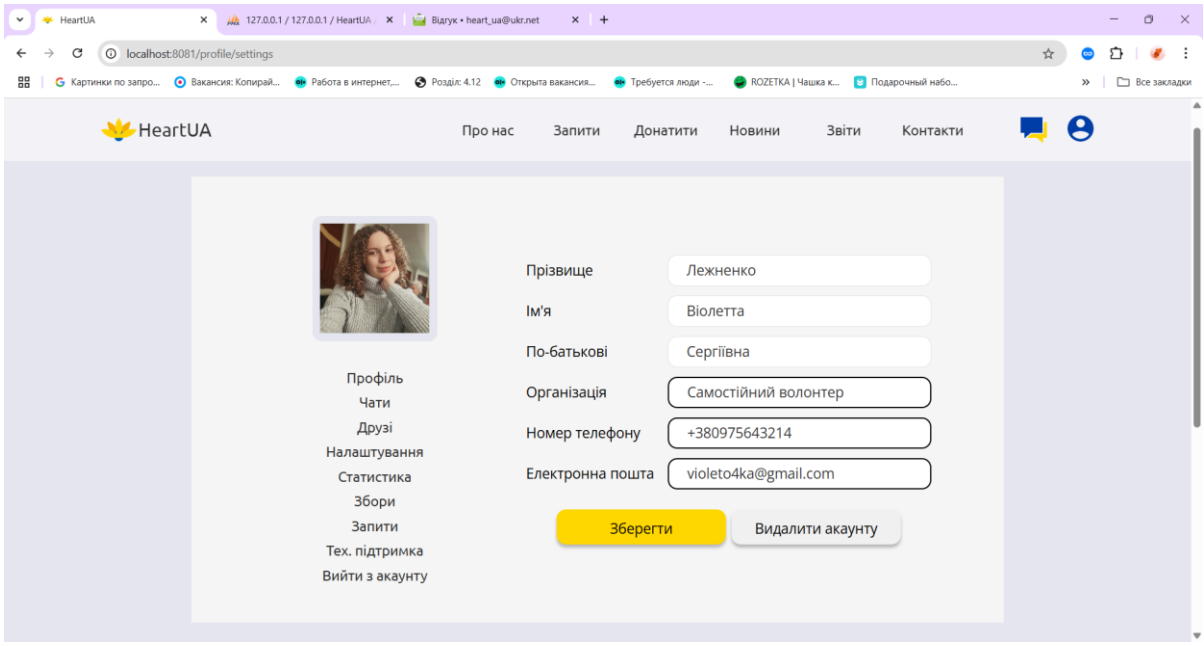


Рисунок 4.29 – Налаштування профілю

Для відслідковування зміни даних на сторінці використано двостороннє зв’язування даних за допомогою v-model.

Для реалізації видалення облікового запису використано велику кількість SQL запитів. Спершу перевірено чи у користувача наявні створені і незавершені збори чи запити, потім, у разі наявності завершених зборів та запитів, перевірено, чи у всіх завершених зборів та запитів наявні заповнені звіти. І тільки якщо все заповнено то тільки тоді видаляється дані про користувача з усіх таблиць БД.

Підсторінка «Друзі» (рис. 4.30) створена за подобою структури сторінок «Запити», «Новини», тощо, тобто за тим самим принципом мотрійки.

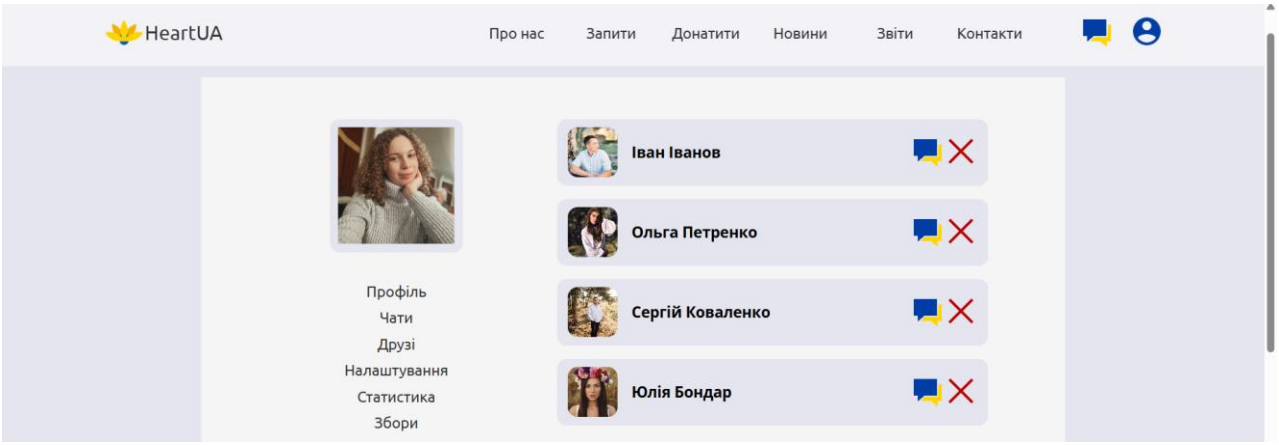


Рисунок 4.30 – Підсторінка «Друзі» у профілі користувача

Для реалізації видалення користувача зі списку друзів застосовано декілька SQL запитів, описаних наступним програмним кодом (рис. 4.31).

```
$sql = "DELETE FROM Friends WHERE idUser = :idU AND idFriend=:idF";
$stmt = $dbcon->prepare(query: $sql);
$stmt->bindParam(param: ':idU', var: &$idU, type: PDO::PARAM_INT);
$stmt->bindParam(param: ':idF', var: &$idF, type: PDO::PARAM_INT);
$stmt->execute();

$stmt = $dbcon->prepare(query: "SELECT numFriends FROM Users WHERE id=:idU");
$stmt->execute(params: [':idU' => $idU]);
$dev = $stmt->fetch(mode: PDO::FETCH_ASSOC);
$nF = $dev['numFriends'] - 1;
$stmt = $dbcon->prepare(query: "UPDATE Users SET numFriends =:nF WHERE id=:idU");
$stmt->execute(params: [':nF' => $nF, ':idU' => $idU]);
```

Рисунок 4.31 – Запит до БД на видалення друга

Тобто спочатку видаляється користувач з таблиці Friends, а далі з БД отримуємо кількість друзів поточного користувача, зменшуємо цю кількість на 1 та оновлюємо це поле.

Підсторінки «Збори» (рис. 4.32) та «Запити» організовані подібним чином, містять у собі списки відповідних компонентів.

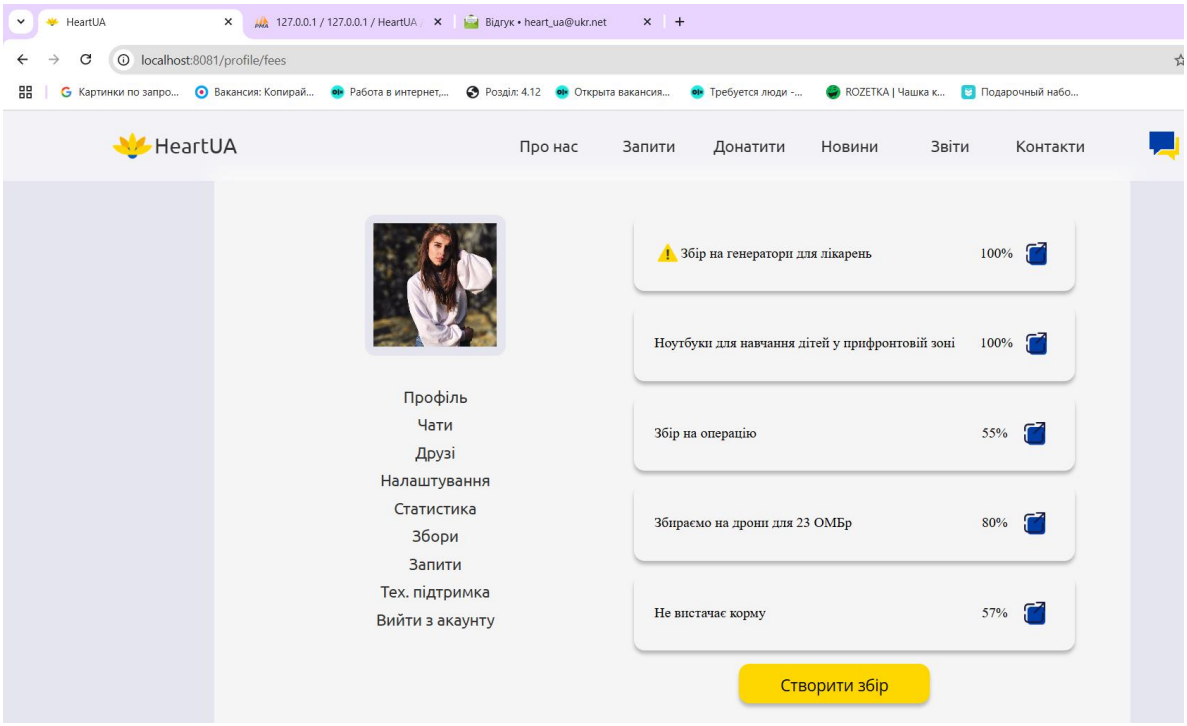


Рисунок 4.32 – Підсторінка профілю «Збори»

На сервер відправляється SQL запит, який, наприклад, у випадку запитів описано наступним програмним кодом (рис. 4.33).

```
$sql = "
    SELECT
        req.*,
        r.id AS r_id,
        r.content AS r_content
    FROM Requests req
    LEFT JOIN Reports r ON req.id = r.idDonate
    WHERE req.idUser = :idUser
";

$stmt = $dbcon->prepare(query: $sql);
$stmt->bindParam(param: ':idUser', var: &$idU, type: PDO::PARAM_INT);
$stmt->execute();
```

Рисунок 4.33 – Отримання даних з БД про запити користувача

Крім простого отримання даних про запит, сервер повертає ще й наявність заповненого звіту до запиту. Це необхідно для коректного відображення у списку картинки трикутника зі знаком оклику.

Сторінка створення нового збору складається з великої кількості обов'язкових полів для вводу (рис. 4.34). Найцікавішими з них є поля для вибору картинки та обкладинки збору. Для цього, у JS документі створено декілька змінних, необхідних для зберігання картинки та посилання на обрану картинку у тимчасовому сховищі. Коли картинка не обрана, то на екрані буде input з атрибутом type="file", а коли картинка вже обрана, вона зберігається у першій змінній, а у другій, за допомогою функції URL.createObjectURL створюється те саме посилання на тимчасове сховище, яке надалі використано у HTML документі сторінки, реалізувавши таким чином перегляд обраного файлу зображення.

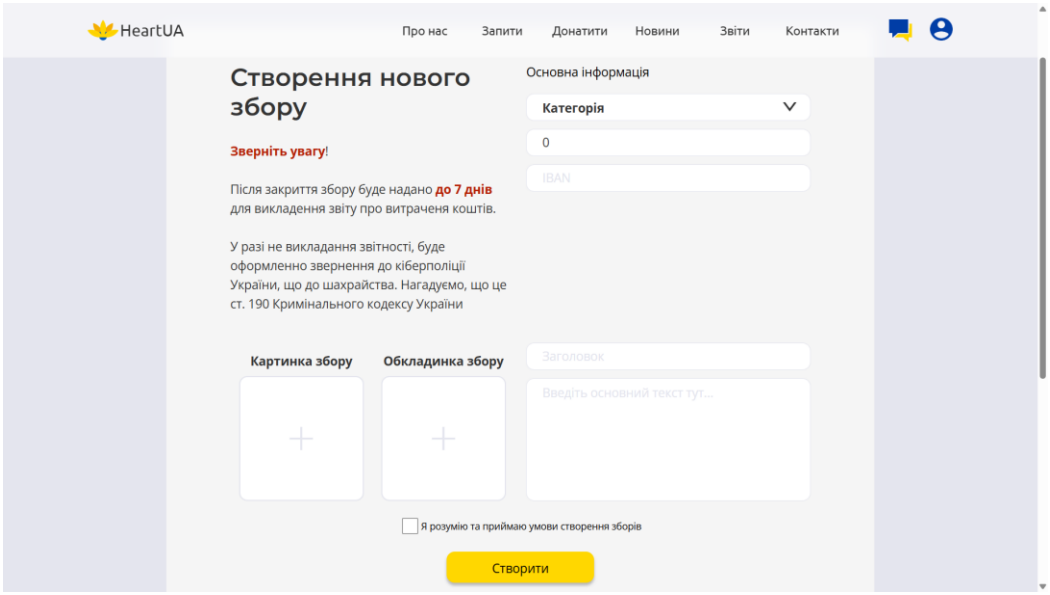


Рисунок 4.34 – Сторінка створення нового збору

Функція fetchNewMFInfo() передає дані про новий збір за допомогою функції fetch(), попередньо провівши перевірку всіх полів форми та зібравши данні у один об'єкт FormData(). Такий підхід необхідний у випадку, коли передавати треба не тільки текст або число, а й файли. Після отримання даних, сервер використовує SQL запит INSERT для додання всієї основної інформації у БД. А от ситуація з картинками трішки відрізняється, адже саму картинку у БД зберігати буде недоцільно, тому у БД зберігається лише назва зображення, у якій є число в кінці,

що означає id збору. А саме зображення зберігатися буде у папці на сервері, тому використано функцію move_uploaded_file(), якому передаємо назву файлу з тимчасового сховища, шлях до папки, куди зберігати файл, та назву зображення. Програмний код реалізації відображено у додатку Г.

Сторінка створення нового запиту розроблена аналогічно сторінці створення збору, але з певними особливостям, а саме іншим набором даних, необхідних для створення.

Структура сторінки редагування даних збору (рис. 4.28) подібна до структури раніше згаданої сторінки створення зборів, але під час першого відображення сторінки робиться запит на сервер для отримання по id даних про збір та за допомогою v-model діє відображення цих даних та слідкування їх змін. На рисунку 4.35 відображено також, що збір закрито. Таке повідомлення з кнопкою з'являється у випадку коли збір (або запит), закрито на 100%. Умови для відображення тих чи інших компонентів застосовані за допомогою директиви v-if в html компоненту.

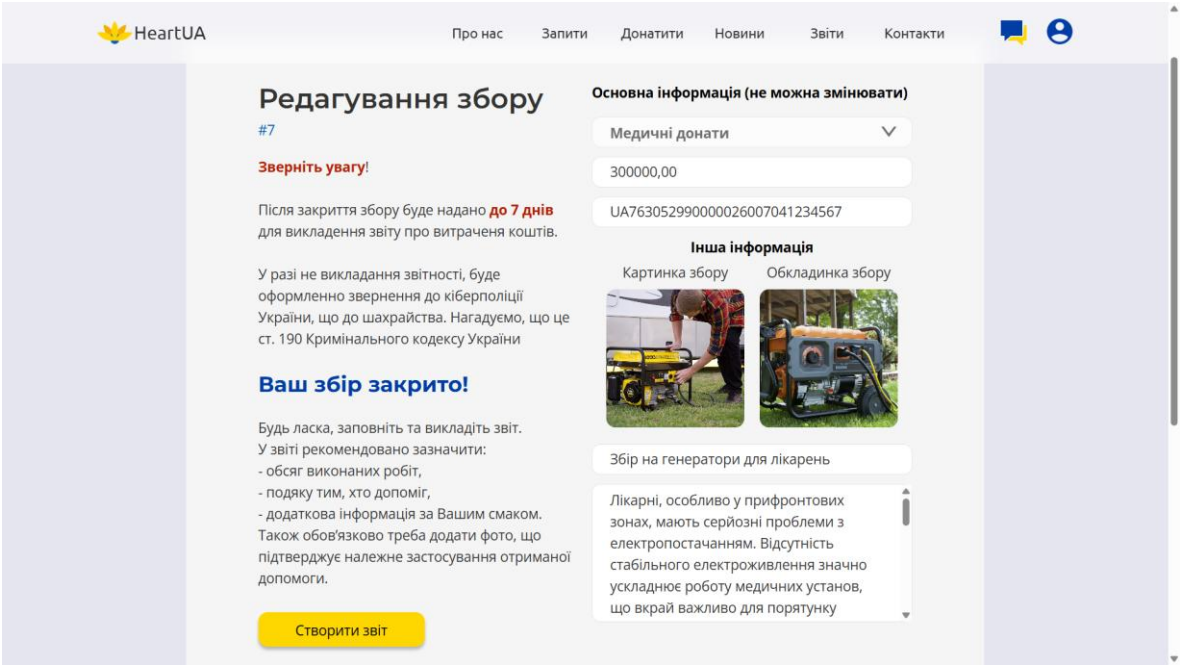


Рисунок 4.35 – Сторінка редагування збору

У випадку, коли кнопка «Створити звіт» показана користувачу, то при натисканні на неї з’явиться форма для заповнення даних звіту (рис. 4.36).

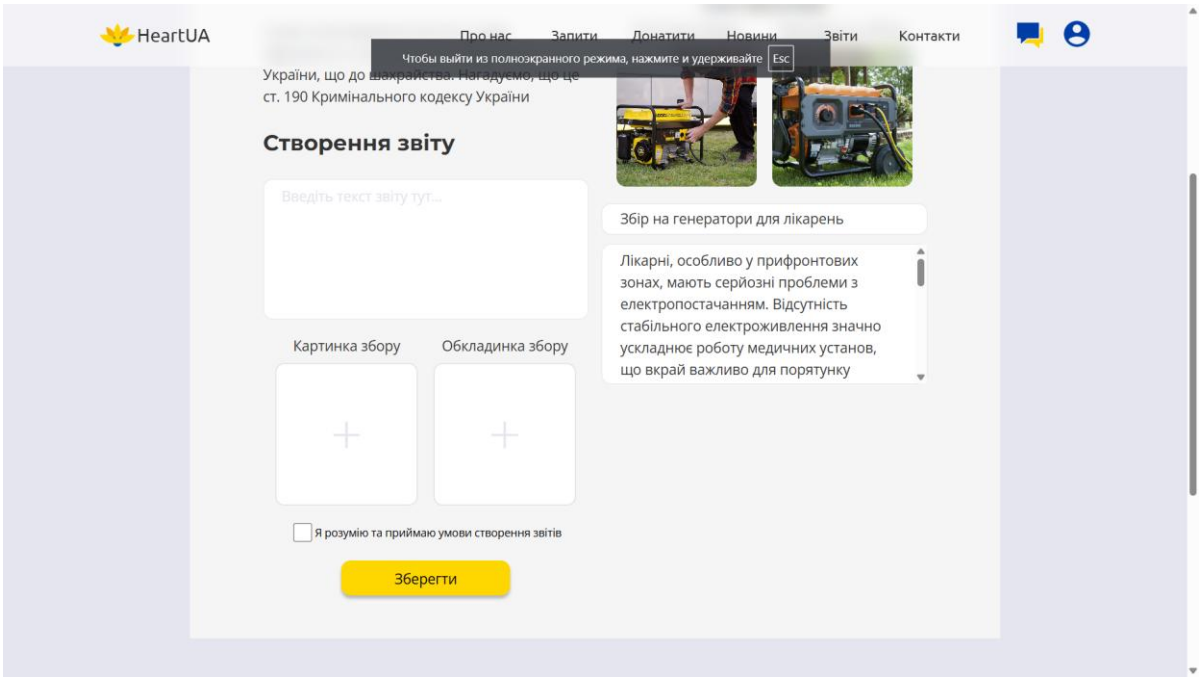


Рисунок 4.36 – Форма для створення звіту

Для зберігання звіту використано SQL запит, опитаний наступним програмним кодом (рис. 4.37).

```
$sql = "UPDATE Reports SET content=:content, isActive=1, image=:image, picture=:picture
WHERE id = :idR";
$stmt = $dbcon->prepare(query: $sql);
$stmt->bindParam(param: ':idR', var: &$idR, type: PDO::PARAM_INT);
$stmt->bindParam(param: ':content', var: &$content, type: PDO::PARAM_STR);
$stmt->bindParam(param: ':image', var: &$imageName, type: PDO::PARAM_STR);
$stmt->bindParam(param: ':picture', var: &$pictureName, type: PDO::PARAM_STR);
$stmt->execute();
```

Рисунок 4.37 – Запит для зберігання інформації звіту

Тобто звітне створюється а оновлюється, а створюється він пустим на етапі, коли збір чи запит закривається. Зміна картинок відбувається таким самим чином, як і зберігання вперше, за допомогою функції move_uploaded_file().

Редагування звітності реалізовано таким самим чином, як і створення, тому не підлягає детальному опису.

Цікавою складової сторінки редагування запиту є список з пропозиціями (рис. 4.38). Наприклад, якщо це збір матеріалів, то у пропозиціях буде видно користувача, кількість яку він пропонує та кнопки «Підтвердити», «Написати» та «Відмовитися».

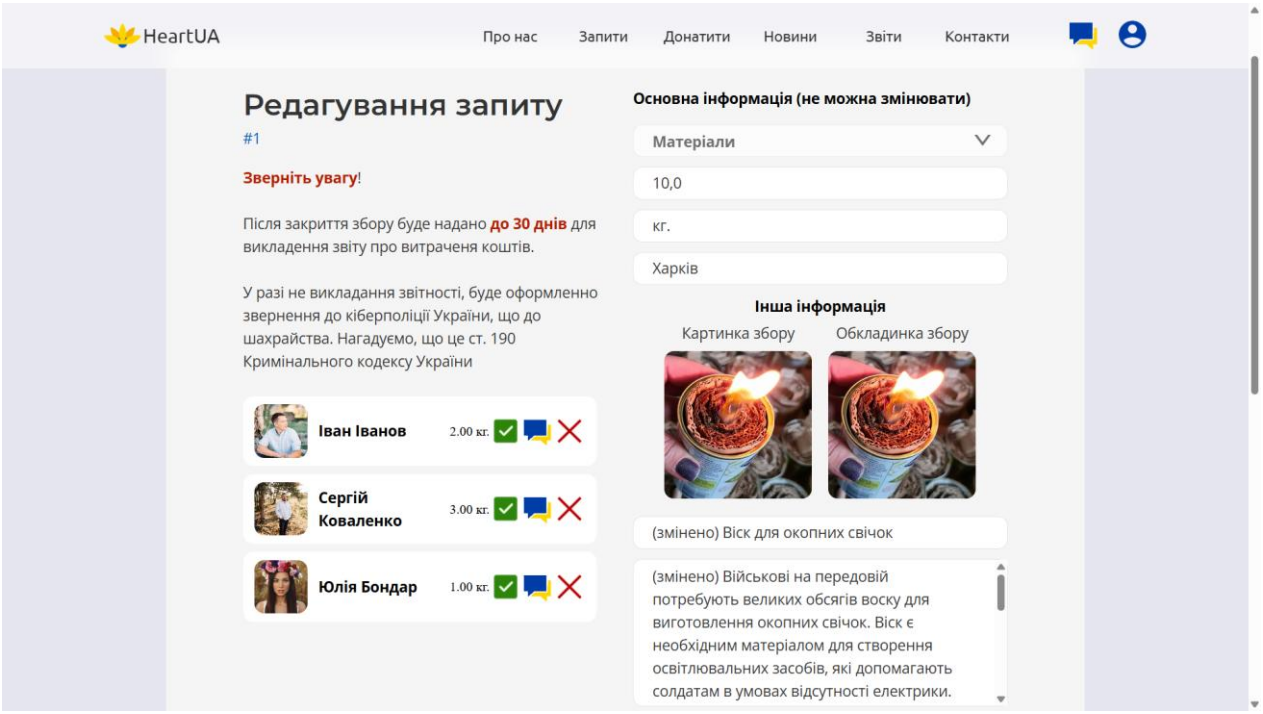


Рисунок 4.38 – Список запропонованої допомоги у запиті

Створено єдиний код на прийняття пропозиції допомоги у запиті для різних типів запитів. На стороні сервера оновлюються дані запиту при прийнятті допомоги. У випадку з матеріалами до поля amount додається кількість матеріалу, якою поділився інший користувач. Також PHP код, вказаний у додатку Г, перевіряє, чи заповнено запит на 100%. У випадку істинності даного твердження інші пропозиції буде видалено і на екрані з’явиться пропозиція створити звіт.

У запиту «Порада» список порад (рис. 4.39) має, відмінний від попереднього, вигляд, адже складається з елементів для показу інформації про користувача та

textarea з встановленою властивістю readonly. Для такого запиту немає необхідності викладати звіт, тож список порад просто необхідний для користувача, і у випадку бажання видалити якесь з них, на сервер відправляються дані про пораду і видаляють запис про пораду за допомогою SQL запиту.

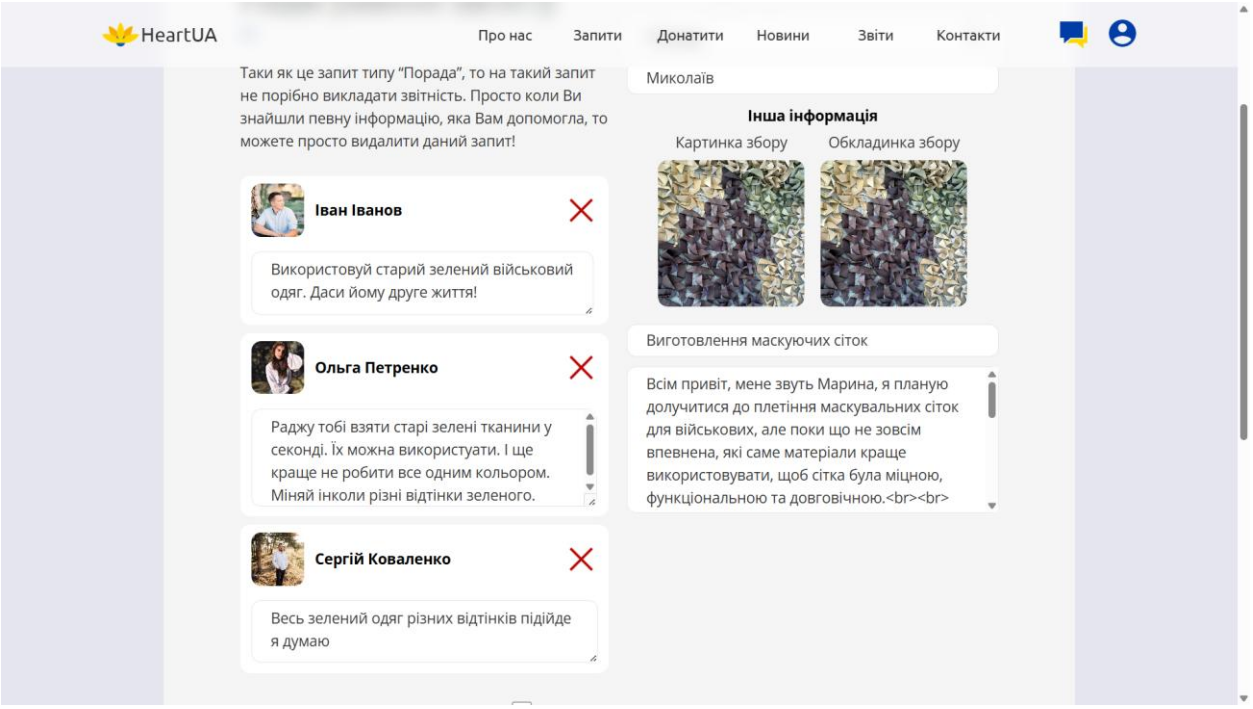


Рисунок 4.39 – Список порад на сторінці редагування запиту типу «Порада»

Через великий об’єм коду у цьому розділі розглянуто всі основні моменти, а повний програмний код вебзастосунку можна подивитися детально на платформі GitHub [35].

4.2 Керівництво користувача

Так як вебзастосунок створений для управління волонтерським ресурсами та взаємодії з благодійними організаціями, тож користувач може виконувати на сайті доволі багато дій (додаток Д).

Процес авторизації користувача розділено на декілька важливих етапи. Основний сценарій відображено на рисунку 4.40. На сторінку авторизації користувач потрапляє відразу після переходу на платформу. Далі заповнити

потрібно такі поля, як логін та пароль. Після натискання кнопки «Увійти» відбувається перевірка коректності введених даних, та, якщо все вірно, користувача перенаправить до головної сторінки.

Рисунок 4.40 – Процес авторизації користувача

Альтернативний сценарій припускає, що користувач може невірно ввести якісь дані, тому у випадку, коли у БД не знайдеться логіну з таким паролем, користувач побавить повідомлення про невірність введених даних.

Основний сценарій процесу реєстрації користувача зображено на рисунку 4.41. Але є декілька альтернативних варіантів, зокрема, якщо користувач введе не всі поля, або якесь поле введе неналежного формату (наприклад, номер телефону, або електронну пошту), або не відмітить згоду на обробку персональних даних, то користувач побачить відповідне повідомлення та реєстрація не відбудеться. Також Зареєструватися у користувача не вийде у випадку, якщо він буде молодше 18 років, тож з’явиться повідомлення про те, що недосягнуто відповідного віку для реєстрації.

Рисунок 4.41 – Основний сценарій реєстрації користувача

У випадку, якщо користувач не хоче реєструватися на платформі, він може зайти у якості гостя, натиснувши відповідно кнопку на сторінці авторизації. Але у

такому випадку, на відміну від зареєстрованого користувача, у нього не буде доступу до особистого кабінету, він не зможе допомагати у запитах та використовувати внутрішній чат платформи для спілкування, але зможе здійснювати благодійні грошові перекази на підтримку проєкту або певні збори, ознайомитися з новинами та звітами.

Для того, щоб видалити обліковий запис треба притримуватися такого сценарію, відображеного у діаграмі на рисунку 4.42.

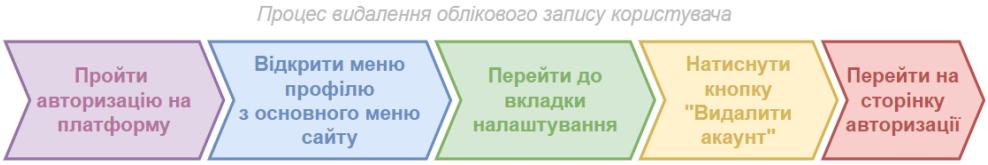


Рисунок 4.42 – Сценарій видалення облікового запиту користувача

Альтернативних сценаріїв видалення облікового запиту може бути декілька. По перше, якщо користувач має незакриті збори чи запити, то видалити не вийде і він побачить відповідне повідомлення. Якщо ж у користувача будуть закриті збори та запити, але не будуть до них заповнені звіти, то видалити акаунт також не вдасться. Тому у випадку незакритих зборів та запитів їх треба закривати і заповнювати всі необхідні звіти по ним.

Налаштовувати профіль треба кожному користувачу. Наразі можливо змінювати організацію, якщо користувач працює на благодійну організацію, номер телефону та електронну адресу. Основна інструкція зміни інформації профілю відображена на рисунку 4.43.

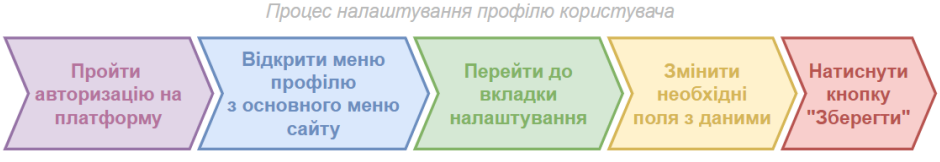


Рисунок 4.43 – Основний сценарій налаштування профілю

Так як програма перевіряє правильний формат номеру телефону та електронної пошти, то у випадку помилки користувач побачить спливаюче вікно з повідомленням.

Якщо користувач бажає змінити картинку свого облікового запису, тоді йому слід діяти інакше (рис. 4.44).

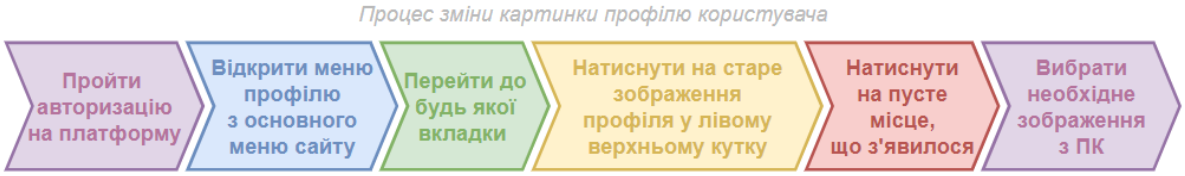


Рисунок 4.44 – Дії, необхідні для зміни зображення профілю

У випадку якщо користувач просто скине старе зображення, а нове не обере, то просто наступного разу, коли він перейде у профіль то буде видно старе зображення.

Спілкування є одним з найважливішим інструментом комунікації між людьми, тож, щоб спілкуватися з іншими користувачами треба їм написати.

У випадку, якщо з користувачем вже є чат з повідомленнями, тоді достатньо відкрити список чатів і натиснути на потрібний, потім написати повідомлення у поле, розташоване у нижній частині чату та натиснути кнопку «Відправити».

У разі відсутності чату з якимось користувачем, тоді треба перейти на сторінку необхідного користувача та натиснути кнопку «Написати» і відкриється чат де можна відправити повідомлення.

Є декілька шляхів відкриття сторінки іншого користувача, серед них такі:

– натиснувши на показник «ID користувача» на сторінці конкретного запиту;

– натиснувши на показник «ID користувача» на сторінці конкретного звіту;

– натиснувши на фотографію користувача у чаті;

– натиснувши на фотографію користувача у списку друзів.

Крім того, написати можна і друзям. Для цього треба притримуватися послідовності дій, відображених на рисунку 4.45.

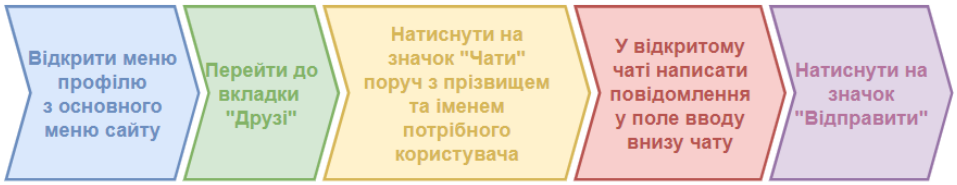


Рисунок 4.45 – Інструкція написання повідомлення другу

Щоб додати користувача у друзі необхідно перейти на сторінку користувача та натиснути кнопку «Додати у друзі», після чого з’явиться кнопка «Видалити з друзів». Це означає, що користувача додано до списку друзів. У випадку, якщо натиснути кнопку «Видалити з друзів», то кнопка змінить назву на «Додати у друзі» та користувача буде вилучено зі списку ваших друзів.

Щоб поскаржитися на іншого користувача, достатньо перейти на його сторінку та натиснути кнопку «Поскаржитися» і поточного користувача буде перенаправлено на сторінку «Контакти», де можна заповнити форму, де вказати скаргу на користувача.

Зв’язатися з технічною підтримкою можна на сторінці «Контакти», перейшовши на неї, використавши головне меню сайту. Далі можна заповнити форму звернення всіма необхідними даними та натиснути кнопку «Відправити». Альтернативним сценарієм зв’язку з технічною підтримкою є заповнення не всіх полів форми, що призведе до появи відповідного повідомлення користувачу.

Для того, щоб переглянути останні 3 новини, достатньо перейти на головну сторінку та прокрутити її до блока «Останні новини». Якщо користувач хоче переглянути усі новини, то йому слід скористатися головним меню сайту та натиснути «Новини» і він перейде до сторінки сторінок, де буде зображено картки новин з короткою інформацією. Щоб Подивитися детально інформація про якусь

конкретну новину, достатньо на відповідній картці новини натиснути кнопку «Дізнатися більше».

Для ознайомлення з відкритими зборами інших користувачів достатньо перейти на сторінку за зборами, натиснувши кнопку «Донатити» у головному меню сайту.

Для того, щоб побачити детальну інформацію про збір достатньо натиснути кнопку «Донатити» картки конкретного збору зі списку зборів.

Для здійснення грошового переказу на певний збір, треба дотримуватися такої послідовності дій, вказаних схемою на рисунку 4.46.

Рисунок 4.46 – Інструкція здійснення грошового переказу на певний конкретний збір

Для здійснення пошуку необхідного збору у поле пошуку на сторінці зі списком зборів можна ввести або заголовок збору, або його унікальний ідентифікатор. Альтернативний сценарій даного процесу може бути у випадку, коли жодних варіантів не буде знайдено. У цьому випадку користувач просто не побачить на екрані жодного варіанту, який міг би підійти.

Для фільтрація зборів можна використати 2 випадаючих списки, які знаходяться поруч з полем для пошуку. Щоб застосувати фільтри треба лише обрати варіант зі списків.

Для створення власного збору треба притримуватися основного сценарію, зображеного на рисунку 4.47.

Рисунок 4.47 – Основний сценарій створення власного збору

У випадку, коли користувач введе не всі дані або певні з них будуть невірного формату, то він побачить спливаюче вікно з підказкою про невірність введених даних.

Для редагування власного збору інструкція подібна, але з певними особливостями, відображеними на рисунку 4.48.

Рисунок 4.48 – Поетапна інструкція редагування власного збору

В даному випадку, у разі невірно введених даних користувач отримає повідомлення з підказками.

Для видалення власного збору необхідно перейти на сторінку редагування збору, який хочете видалити, і натиснути кнопку «Видалити збір». Але є нюанс у тому, що якщо збір вже розпочато, тобто хтось вже на цей збір переказував кошти, тоді збір видалити не вдасться. Видалити можливо лише за умови, що на збір ще ніхто не жертвував кошти.

З запитами сценарії подібні до зборів. Для створення власного запиту необхідно перейти у профіль у вкладку «Запити» та натиснути «Створити». Для редагування запиту треба у вкладці «Запити» профілю навпроти конкретного запиту натиснути синій квадратик та перейти на сторінку редагування, де вже можна змінити інформацію про запит або взагалі видалити його (якщо ще ніхто не

допомагав закрити запит, але при цьому запит типу Порада можна видалити у будь який момент). Пошук, фільтрація та ознайомлення з конкретним запитом відбувається аналогічно до зборів, тальки на сторінці «Запити».

А от процес пропонування допомоги у запитах має певні особливості, пов'язані з тим, що запити є трьох різних видів. Основний сценарій вказано у вигляді схеми на рисунку 4.49.

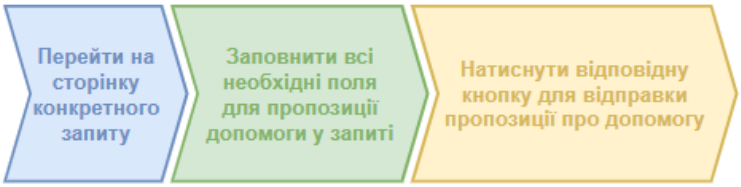


Рисунок 4.49 – Основний сценарій пропозиції допомоги у запиті

Якщо запит має тип матеріали, то на сторінці конкретного запиту треба у спеціальне поле ввести кількість того матеріалу, яким може поділитися поточний користувач та натиснути на кнопку «Запропонувати». Якщо це запит типу «Фіз. допомога», тоді достатньо просто натиснути на кнопку «Записатися», а от якщо це запит типу «Порада», то у спеціальне поле для вводу треба ввести текст поради та натиснути «Відправити».

У випадку, якщо на Ваш запит відгукнулися, тоді на сторінці редагування запиту можна побачити перелік пропозиції допомоги. У ситуації з запитом типу «Матеріали», то таку пропозицію можна відхилити, натиснувши на червоний хрестик, можна написати іншому користувачу, договоритися про передачу матеріалів, та підтвердити допомогу, коли матеріали успішно передані. Якщо говорити про фізичну допомогу, то можна видалити пропозицію, або написати користувачу та договоритися про фізичну допомогу, та коли той користувач допоміг підтвердити допомогу. Якщо запит має тип «Порада», тоді список пропозицій буде складатися просто з порад інших користувачів, які можна прочитати та видалити.

Звіти необхідні для досягнення прозорості роботи вебзастосунку та контролю застосування благодійної допомоги.

Для створення звіту треба скористатися такою послідовністю дій, зображених на рисунку 4.50.

Рисунок 4.50 – Покрокова інструкція створення звіту

Редагувати звіт можна таким самим чином, перейшовши на сторінку редагування запиту чи збору, натиснувши кнопку «Редагувати звіт», відредагувавши необхідну інформацію та натиснувши кнопку "Зберегти".

Для того, щоб ознайомитися зі звітами інших користувачів достатньо перейти на сторінку «Звіти», скориставшись головним меню сайту. Якщо ж треба подивитися детальну інформацію на конкретний звіт, то можна натиснути на кнопку «Читати звіт». У випадку якщо інший користувач ще не заповнив звіт, то на кнопку не можна буде натиснути.

Знайти конкретний звіт можна за його id або назви, використавши поле пошуку на сторінці «Звіти», просто набравши текстом, що бажаєте знайти.

4.3 Тестування

Метою тестування є перевірка коректності роботи вебзастосунку, відповідності вимогам, зазначеним у постанові задачі, знаходження та виправлення помилок у разі їх виникнення та забезпечення зручного користування системою.

Першочергово визначено тест-кейси та проведено функціональне тестування, тобто перевірено роботу всіх основних функції, таких як авторизація та реєстрація користувачів, робота чату, функції пошуку та фільтрації запитів та зборів, відправки листа у технічну підтримку, тощо. Також перевірено відповідність вебплатформи

функціональним вимогам. Приклади тест-кейсів для форми реєстрації нового користувача вказано у таблиці 4.1.

Таблиця 4.1 – Приклади тест-кейсів для тестування форми реєстрації користувача

1	Користувач на сторінці авторизації натискає кнопку «Реєстрація та потрапляє на сторінку Реєстрації»	Користувач вводить всі необхідні данні, але номер телефону введено не у форматі «+380XXXXXXXXX», погоджується на обробку персональних даних та натискає на кнопку «Реєстрація»	Користувач бачить на екрані повідомлення «Не вірний номер телефону. Спробуйте ще раз! Приклад: +380XXXXXXXXX»	Користувач бачить на екрані повідомлення «Не вірний номер телефону. Спробуйте ще раз! Приклад: +380XXXXXXXXX»	Пройдено
2	Користувач на сторінці авторизації натискає кнопку «Реєстрація та потрапляє на сторінку Реєстрації»	Користувач вводить всі необхідні данні, але електронну пошту введено не у форматі «local-part@domain», погоджується на обробку персональних даних та натискає на кнопку «Реєстрація»	Користувач бачить на екрані повідомлення «Не вірний формат електронної пошти»	Користувач бачить на екрані повідомлення «Не вірний формат електронної пошти»	Пройдено
3	Користувач на сторінці авторизації натискає кнопку «Реєстрація та потрапляє на сторінку Реєстрації»	Користувач вводить всі необхідні данні, погоджується на обробку персональних даних та натискає на кнопку «Реєстрація», але пароль не задовольняє наступним вимогам.	Користувач бачить повідомлення «Ненадійний пароль, будь ласка, мають виконуватися всі умови:» та список вимог до паролю.	Користувач бачить повідомлення «Ненадійний пароль, будь ласка, мають виконуватися всі умови:» та список вимог до паролю.	Пройдено

Тестування інтерфейсу (UI) проведено шляхом порівняння реалізованого вебзастосунку [35] з дизайн документом [34]. Також перевірено роботу навігації на сайті, правильність та зручність переходів між сторінками.

Кросбраузерне тестування допомогло знайти деяку неточність у програмному коді. Перевірка роботи вебзастосунку відбувалася за допомогою браузерів Google Chrome, Mozilla Firefox, Opera, Web Explorer та Microsoft Edge. На усіх браузерах, крім Mozilla Firefox, вебзастосунок не втратив стилі, працював правильно і без зупинок, а ось у Mozilla Firefox стилі індикатору виконання частково не завантажилися. Порівняти рівницю можна на рисунку 4.51, де зліва відображено індикатор виконання, запланований дизайн документом та відображений у браузері Google Chrome, а праворуч те, як його відобразив Mozilla Firefox.

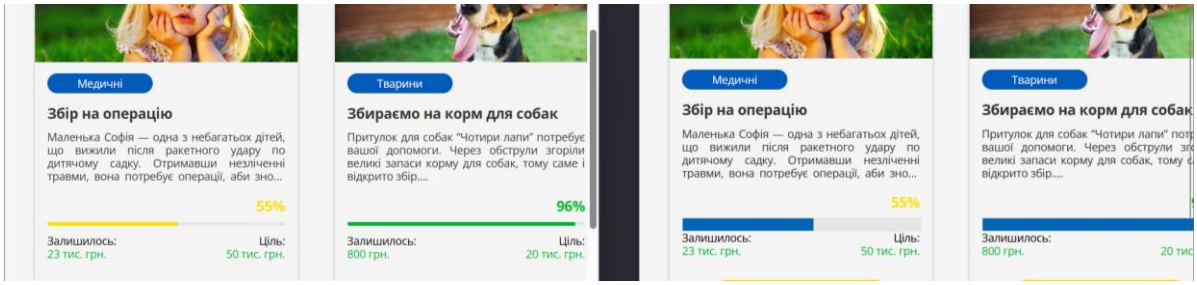


Рисунок 4.51 – Відмінність індикатора виконання на різних браузерах

Так, як браузер Mozilla Firefox не підтримує стилізацію самого індикатора, то прийнято рішення заміни тегу <progress> на звичайний контейнер . В результаті вийшов наступний програмний код (рис. 4.52).

```
<div class="custom-progress">
  <div
    class="progress-fill"
    :class="percent < 21 ? 'progress-low' : percent < 76 ? 'progress-medium' : 'progress-high'"
    :style="{ width: percent + '%' }"
  ></div>
</div>
```

Рисунок 4.52 Оновлений контейнер для індикатора виконання

Стилі задано відносно вже до контейнеру, а показ відповідно отриманого відсотка здійснюється за допомогою атрибуту style у якому вказано, що ширина контейнеру дорівнює відсотку, тож тепер цей елемент відображається однаково на різних браузерах (рис. 4.53).

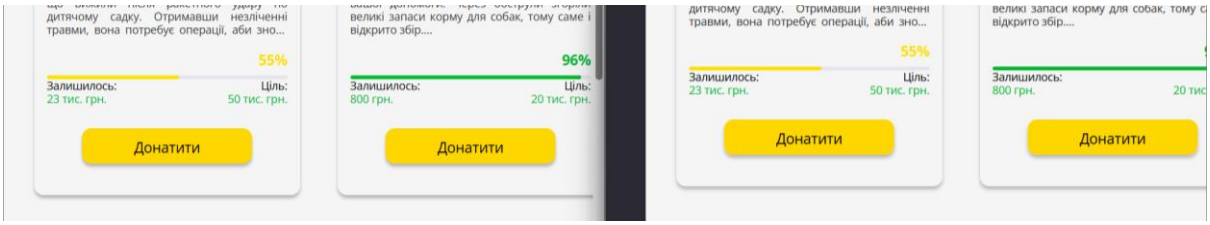


Рисунок 4.53 – Результат після зміни підходу до візуалізації індикатора виконання

Висновки до розділу 4

Вебзастосунок «HeartUA» реалізовано за клієнт-серверною архітектурою, причому клієнтська частина написана з використанням фреймворку Vue.js та мовою програмування JavaScript, а серверну частину розроблено на PHP. Обмін даними між клієнтською та серверною частиною відбувається через REST API. Повний код вебзастосунку представлено на GitHub [35]. Програмно реалізовано та виконано всі вимоги, описані у постановці задачі КР.

Керівництво користувача представляє інструкцію по використанню вебсервісу, описує можливості користувача та можливі нюанси використання платформи.

Як у процесі самої розробки так і після завершення програмної реалізації проведено різни типи тестування вебсистеми, зокрема кросбраузерне, функціональне тестування, тестування користувацького інтерфейсу (UI) та UX-тестування. Процес перевірки програмного застосунку дозволив визначити та покращити вебсистему для забезпечення її коректної роботи.

5 ПЕРСПЕКТИВИ РОЗВИТКУ ВЕБЗАСТОСУНКУ ПІДТРИМКИ ВОЛОНТЕРСЬКОЇ ДІЯЛЬНОСТІ

5.1 Функціональне розширення системи

Для покращення роботи вебзастосунку, можна реалізувати рейтинг волонтерів, залежно від кількості їх допомоги у закритті зборів та запитів. Крім того можна додати накопичення певних віртуальних одиниць для обміну їх на товари спонсорів проєкту. Таким чином люди ще більше будуть замотивовані приєднуватися до платформи, а спонсори отримають сильну рекламу та підвищення рівня впізнаваності бренду.

Реалізувати окремий вхід для адміністратора та модераторі буде гарним рішенням для покращення управління вебплатформою. Система вже буде поділятися на 2 частини, одна з яких та, яку бачать звичайні користувачі, а інша – для адміністраторів та модераторів. Другу частину можна створити під потреби конкретного спеціаліста, наприклад для адміністратора це буде статистика сайту та задання певних обмежень користувачам, а для модератора – список скарг користувачів та можливість блокування користувачів, перевірки їх інформації, тощо.

У разі розширення вебзастосунку на закордоння, то реалізація зміни мови сайту було б дуже доречно, причому додати саме список вибору мови на сайті.

Для покращення комунікації можна розширити можливості чату, а саме: додати можливість відправлять картинки, документи, змінювати стиль тексту та додавати реакції на повідомлення. Це створить відчуття більш «живого» спілкування. Крім того можна додати можливість створення групових чатів.

Розбиття користувачів на самостійних волонтерів та волонтерів при організаціях дало б можливість більш чітко розмежовувати людей та їх запити. На поточному етапі проєкту волонтер може зареєструватися та додати у себе в профілі належність до якоїсь організації, але створення та реєстрація самої організації призвело б до підвищення рівня довіри користувачів.

Крім усього вище зазначеного було б непогано автоматизувати процес контролю за викладенням звітів, реалізувавши планувальник завдань, який сповістить користувача про те, що закритий звіт чи запит потребує написання звіту, та після завершення терміну очікування надсилає попередження про те, що якщо звіт не буде викладено, то дані будуть передані до поліції.

5.2 Інтеграція з зовнішніми сервісами

На поточному етапі розвитку вебзастосунку реалізовано лише тестову версію системи оплати з використанням платіжної системи Fondy, яка імітує перекази коштів, але не робить їх явно. Підключення повної версії було б обов’язковою умовою у разі подальшої публікації вебплатформи в інтернет просторі. Крім того, для зручності користувачів можна було б реалізувати інтеграцію з іншими платіжними системами, на кшталт PayPal, Apple Pay, тощо.

Додатково можна було б реалізувати інтеграцію з системами електронної ідентифікації, до яких відноситься всім відома Дія. Такий підхід здатний значно пришвидшити процес реєстрації нових користувачів.

Варто й згадати про можливість інтеграції з email розсилками. Вони здатні підвищити обізнаність користувачів у новинах благодійної сфери, а також будуть слугувати нагадуванням про активність на платформі.

5.3 Модернізація безпеки даних

Якщо повернутися до пропозиції інтеграції сервісу Дія, то варто зазначити, що крім пришвидшення процесу реєстрації нових користувачів, такий підхід підвищить безпеку, наприклад якщо інтегрувати Дія Антибот, тоді точно можна бути впевненим у тому, що користувач – реальна людина і дані введені точно згідно офіційним документам. Причому це забезпечить безпеку не тільки для самої платформи, але й для інших користувачів.

Крім інтеграції Дії можна ще реалізувати двофакторну автентифікацію. Це забезпечить безпеку користувачів, адже за допомогою підтвердження входу в

обліковий запис ніхто крім господаря облікового запису безпосередньо не зможе увійти у систему.

5.4 Удосконалення UI/UX частини

Для покращення візуальної частини проєкту краще було б додати адаптивну верстку, адже у рамках сьогоденного етапу розвитку проєкту зосередженість була над реалізацією функціоналу системи. Крім того, великим плюсом до престижу проєкту була б реалізація перенаправлення користувача на мобільний застосунок. Це б відкинуло необхідність створення адаптації дизайну під смартфони, але треба було б додатково створювати той самий мобільний застосунок. До того ж, мобільний застосунок ще й дозволить підвищити зацікавленість користувачів до використання вебсистеми, адже наявність повідомлень від застосунка не дасть про себе забути.

Що до покращення візуального сприйняття було б доречно додати можливість зміни режиму на темний. Це приведе до ряду певних покращень, таких як:

- зменшення навантаження на очі користувача у темряві;
- зменшення енергоспоживання пристроїв;
- задоволення власних вподобань користувача (багато людей вважають темний стиль сайту сучаснішим та елегантнішим);
- надання користувачу право вибору (відчуття контролю та персоналізації, що допоможе користувачу відчути себе частиною системи з можливістю її змінювати).

У разі розширення останнього пункту попереднього списку стало б доречним додання налаштувань персоналізації системи у профіль користувача.

Висновки до розділу 5

Не дивлячись на те, що у проєкті повністю реалізовано стандартні функціональні вимоги, вебзастосунок можна безмежно розширювати та

покращувати для подальшого підвищення інтересу громади до даної тематики. На поточному етапі розвитку визначено певний перелік можливих дій для покращення вебсистеми, до яких входять різні функціональні розширення, інтеграції з зовнішніми сервісами, кроки спрямовані на підвищення безпеки та удосконалення візуальної частини платформи.

До функціональних розширень входить модернізація системи комунікації шляхом додавання нового функціоналу у внутрішні чат платформи, реалізація зміни мови сайту, реалізація системи рейтингів користувачів та бонусної системи, розробка системи для керування вебплатформою, тощо.

Крім того, у подальшому важлива інтеграція з іншими різними сторонніми сервісами, на кшталт платіжних систем, систем електронної ідентифікації та поштовими сервісами.

Покращення системи безпеки сприяє підвищенню рівня довіри користувачів до програмного продукту, тож про інтеграцію з Дія та реалізація двофакторної автентифікації не варто забувати.

Удосконалення візуальної складової проєкту допоможе привернути нових користувачів, адже зробить систему більш комфортною, тож у майбутньому можна реалізувати налаштування персоналізації та додати темну тему на сайт.

ВИСНОВКИ

Під час виконання КР створено вебзастосунок для управління волонтерським ресурсами та взаємодії з благодійними організаціями, метою якого є покращення координації між волонтерами та благодійними організація, поширення теми волонтерства та мотивація населення приймати участь у благодійництві, що у свою чергу допоможе ефективніше надавати допомогу людям, які її потребують. Для досягнення поставленої мети виконано певний перелік завдань:

- проведено огляд та аналіз предметної сфери;
- сформовано постановку задачі;
- визначено методи для вирішення завдання;
- проведено аналіз та вибір технологій розробки системи;
- розроблено дизайн документ вебзастосунку;
- створено та налаштовано БД;
- розроблено програмний код вебзастосунку;
- розроблено алгоритм шифрування AES-256-GCM для безпечного зберігання важливих даних про користувача;
- реалізовано інтеграцію з платіжною системою Fondy для забезпечення можливості здійснення грошових переказів на благодійність;
- реалізовано внутрішній чату платформи для забезпечення комунікації;
- проведено тестування основного функціоналу системи;
- визначено шляхів подальшого розвитку проєкту.

Проєкт КР пройшов усі стадії розробки, а саме: проєктування, моделювання, програмна реалізація та тестування. Програмна реалізація вебзастосунку виконує всі поставлені функціональні та нефункціональні вимоги визначені у постановці задачі поточної КР. Тестування надало змогу визначити недоліки програмної реалізації та виправити їх, для покращення цілісної роботи системи.

Визначені перспективи розвитку даного проєкту, до яких входить функціональне розширення системи, додаткові інтеграції з зовнішніми сервісами,

модернізація системи безпеки та удосконалення UI/UX частини, допоможуть у подальшому повністю підготувати проєкт для практичної реалізації поставленої раніше цілі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке благодійний фонд і благодійна організація. Entire : вебсайт. URL: https://entire.com.ua/uk/news/674-shcho-take-blahodiinyi-fond-i-blahodiina-orhanizatsiia.html (дата звернення: 25.04.2025).

2. Про благодійну діяльність та благодійні організації : Закон України від 05.07.2012 № 5073-VI. URL: https://zakon.rada.gov.ua/laws/show/5073-17#Text (дата звернення: 25.04.2025).

3. Благодійна діяльність. Збір благодійних пожертв. WikiLegalAid : вебсайт. URL: https://legalaid.wiki/index.php/Благодійна_діяльність._Збір_благодійних_пожертв (дата звернення: 28.04.2025).

4. Google Forms – Створення форм. Google Workspace : вебсайт. URL: https://workspace.google.com/products/forms/ (дата звернення: 29.04.2025).

5. Миргородський Н. В. Онлайн-платформа зі збору та розподілу гуманітарної допомоги : кваліфікаційна робота ... бакалавра : 126 Інформаційні системи та технології / Київський політехнічний інститут імені Ігоря Сікорського. Київ, 2023. 93 с. URL: https://ela.kpi.ua/handle/123456789/58536 (дата звернення: 28.04.2025).

6. Сергієнко С. В. Веб-орієнтований додаток благодійної організації : кваліфікаційна робота... бакалавра : 121 «Інженерія програмного забезпечення» / Київський національний торговельно-економічний університет. Київ, 2023. 53 с. URL: http://dp.knute.edu.ua/jspui/bitstream/.pdf (дата звернення: 28.04.2025).

7. Пирогов В. І. Розробка веб-застосунку для допомоги волонтерам та внутрішньо переміщеним особам : кваліфікаційна робота ... бакалавра : 121 «Інженерія програмного забезпечення» / Університет митної справи та фінансів. Дніпро, 2023. 64 с. URL: http://biblio.umsf.dp.ua/jspui/handle/123456789/6378 (дата звернення: 29.04.2025).

8. Ghate M. S. Volunteer management system : dissertation ... Master of Science

in Computer Science / California State University, Sacramento. Sacramento, 2011. URL: https://scholars.csus.edu/esploro/outputs/99257830838901671 (дата звернення: 29.04.2025).

9. Андрухович М. А. Веб-застосунок для збору коштів на благодійність : кваліфікаційна робота... бакалавра : 121 «Інженерія програмного забезпечення» / Київський політехнічний інститут імені Ігоря Сікорського. Київ, 2022. 87 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/e990fe29-ba5a-4482-9b14-a50af9bac326/content (дата звернення: 29.04.2025).

10. Платформа добрих справ «Добро.ua». Добро.ua : вебсайт. URL: https://dobro.ua/ (дата звернення: 30.04.2025).

11. Wappalyzer – Технології вебсайтів. Wappalyzer : вебсайт. URL: https://www.wappalyzer.com/ (дата звернення: 30.04.2025).

12. Razom for Ukraine – Головна. Razom for Ukraine : вебсайт. URL: https://www.razomforukraine.org/ua/home_ua/ (дата звернення: 30.04.2025).

13. Метод зважених сум для ефективних рішень. GoIT : блог. URL: https://goit.global/ua/blog/metod-zvazhenykh-sum-dlya-efektyvnykh-rishen/ (дата звернення: 30.04.2025).

14. Методи прийняття управлінських рішень. LivingFO : вебсайт. URL: https://livingfo.com/metody-pryjniattia-upravlinskykh-rishen/ (дата звернення: 30.04.2025).

15. Левіна-Костюк М., Мельничук О., Телічко Н. Методи прийняття управлінських рішень в умовах недостатньої інформації. Економіка та суспільство. 2022. № 43. URL: https://economyandsociety.in.ua/index.php/journal/article/view/1726/1663 (дата звернення: 30.04.2025).

16. What is Advanced Encryption Standard (AES) and How is it Related to NIST? Lazarus Alliance : вебсайт. URL: https://lazarusalliance.com/uk/what-is-advanced-encryption-standard-aes-and-how-is-it-related-to-nist/ (дата звернення: 29.04.2025).

17. Dworkin M. Recommendation for Block Cipher Modes of Operation:

Galois/Counter Mode (GCM) and GMAC : NIST Special Publication 800-38D. Gaithersburg, MD : National Institute of Standards and Technology, 2007. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf (дата звернення: 29.04.2025).

18. JavaScript — DevDocs : інтерактивна документація. URL: https://devdocs.io/javascript/ (дата звернення: 29.04.2025).

19. TypeScript Documentation : офіційний вебсайт. URL: https://www.typescriptlang.org/docs/ (дата звернення: 29.04.2025).

20. Сувора типізація: TypeScript, Flow, JavaScript. IT Hillel : блог. URL: https://blog.ithillel.ua/articles/suvora-typizatsiia-typescript-flow-javascript (дата звернення: 01.05.2025).

21. TypeScript vs JavaScript: Порівняння. Guru99 : вебсайт. URL: https://www.guru99.com/uk/typescript-vs-javascript.html (дата звернення: 01.05.2025).

22. PHP Manual. PHP : офіційна документація. URL: https://www.php.net/manual/uk/ (дата звернення: 01.05.2025).

23. Python Documentation. Python : офіційна документація. URL: https://docs.python.org/uk/3/ (дата звернення: 01.05.2025).

24. Ruby Documentation. Ruby : офіційна документація. URL: https://www.ruby-lang.org/en/documentation/ (дата звернення: 01.05.2025).

25. Vue.js Guide – Introduction. Vue.js : офіційна документація. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення: 01.05.2025).

26. React – Головна сторінка. React : офіційна документація. URL: https://uk.legacy.reactjs.org/ (дата звернення: 01.05.2025).

27. Порівняння фреймворків: Vue.js та React. Habr : стаття. URL: https://habr.com/ru/articles/904698/ (дата звернення: 01.05.2025).

28. Is Vue.js Better Than React? Jelvix : блог. URL: https://jelvix.com/blog/js-frameworks-is-vuejs-better-than-react (дата звернення: 01.05.2025).

29. Visual Studio Code Documentation. Visual Studio Code : офіційна

документація. URL: https://code.visualstudio.com/docs (дата звернення: 01.05.2025).

30. WebStorm – Meet WebStorm. JetBrains : офіційна документація. URL: https://www.jetbrains.com/help/webstorm/meet-webstorm.html?keymap=Windows (дата звернення: 01.05.2025).

31. Sublime Text Documentation. Sublime Text : офіційна документація. URL: https://www.sublimetext.com/docs/index.html (дата звернення: 01.05.2025).

32. Notepad++ User Manual. Notepad++ : офіційна документація. URL: https://npp-user-manual.org/ (дата звернення: 01.05.2025).

33. UI/UX Design: інструменти для прототипування інтерфейсів. URL: (дата звернення: 02.05.2025)

34. Дизайн проєкт вебзастосунку для управління волонтерськими ресурсами та взаємодії з благодійними організаціями. URL: https://www.figma.com/design/LrR9N4plgcu8VOZguOlRqO/HeartUA?node-id=0-1&t=jIGuEgK8Z2dkeWPC-1 (дата звернення: 15.05.2025)

35. Проєкт вебзастосунку для управління волонтерськими ресурсами та взаємодії з благодійними організаціями. URL: https://github.com/Violettalez/HeartUA_Vue.js.git

36. Лежненко В. С. , Кулаковська І. В. Вебзастосунок для управління волонтерськими ресурсами та взаємодії з благодійними організаціями. XXI Міжнародна наукова конференція «Ольвійський форум - 2025: стратегії країн Причорноморського регіону в геополітичному просторі», 16–21червня 2025 року. ЧНУ імені Петра Могили, м. Миколаїв, Україна, СЕКЦІЯ: Технічні науки; Підсекція: Інтелектуальні інформаційні системи.

ДОДАТОК А

Програмний код застосування методів прийняття рішень

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form{
        int E;int Q;
        DataGridViewCell selectedCell = null;
        public Form1(){
            InitializeComponent();
            E = 2;
            Q = 1;
        }
        private void button1_Click(object sender, EventArgs e){
            E = ((int)numericUpDown2.Value);
            Q = ((int)numericUpDown1.Value);
            numericUpDown1.Enabled = false;
            numericUpDown2.Enabled = false;
            dataGridView1.Visible = true;
            richTextBox1.Visible = true;
            label6.Visible = true;
            label7.Visible = true;
            button3.Visible = true;
            button5.Visible = true;
            button4.Visible = true;
            comboBox2.Visible = true;
    richTextBox1.AppendText("Створено матрицю рішень розмірністю " + Q + " на "
+ E + ".\n");
            button1.Enabled = false;
            button5.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            comboBox2.Enabled = false;
            for (int i = 0; i < E; i++){
                dataGridView1.Columns.Add($"E{i + 1}", $"E {i + 1}");
                dataGridView1.Columns[i].Width = 50;
            }
            dataGridView1.Columns.Add($"{E}", "Оптимізація");
            foreach (DataGridViewColumn column in dataGridView1.Columns){
                column.SortMode = DataGridViewColumnSortMode.NotSortable;
            }
            for (int i = 0; i < Q; i++){
                dataGridView1.Rows.Add();
                dataGridView1.Rows[i].HeaderCell.Value = $"Q{i + 1}";
            }
            dataGridView1.AllowUserToDeleteRows = false;
dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        }
        bool AreAllCellsFilled(DataGridView dataGridView){
            foreach (DataGridViewRow row in dataGridView.Rows){
                foreach (DataGridViewCell cell in row.Cells){
```

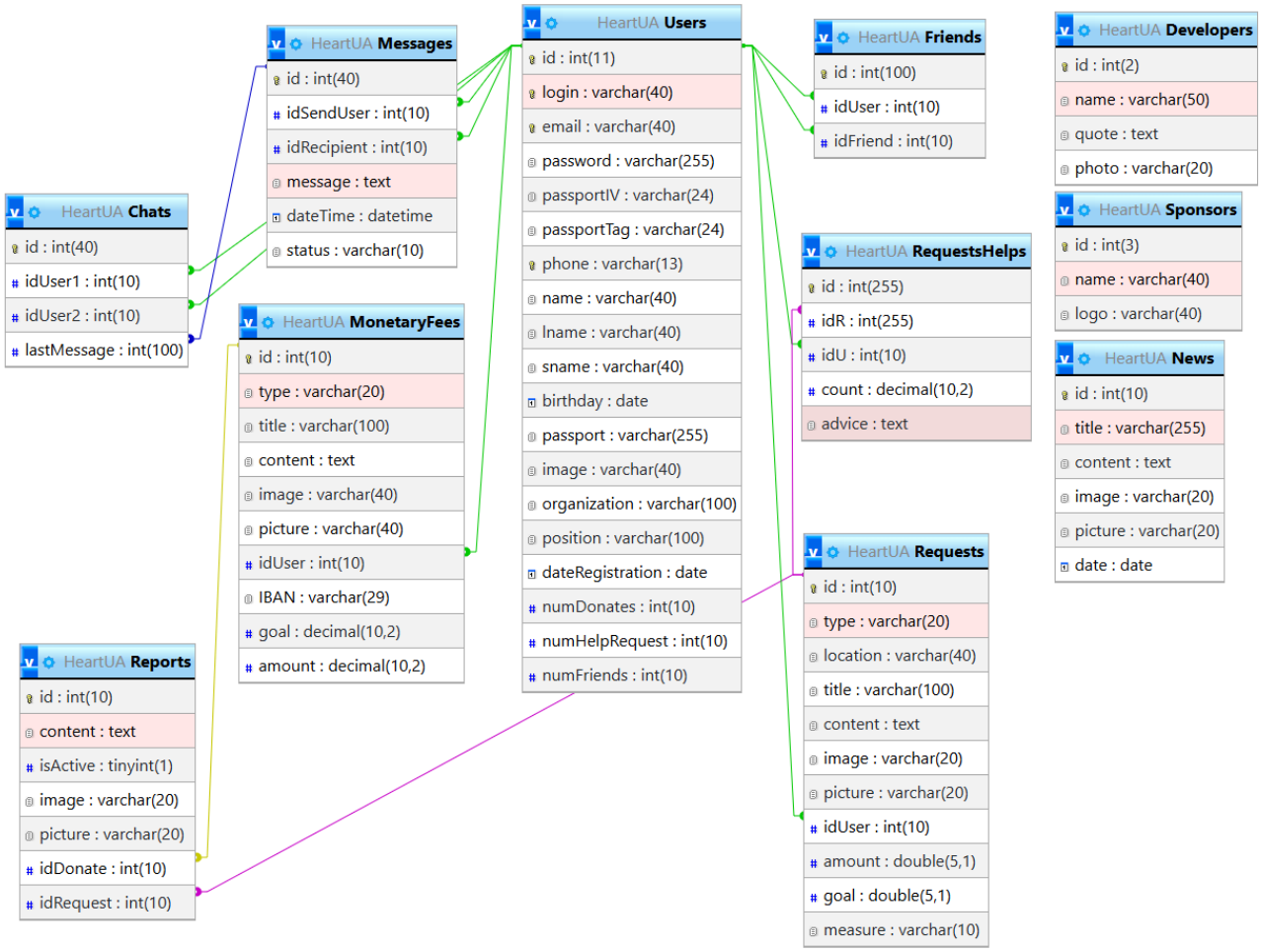
```
                        if (cell.Value == null ||
string.IsNullOrWhiteSpace(cell.Value.ToString())) return false;
                    }
                }
                return true;}
            private void Form1_Load(object sender, EventArgs e)
            {
                dataGridView1.CellValueChanged +=
DataGridView1_CellValueChanged;
                dataGridView1.CurrentCellDirtyStateChanged +=
DataGridView1_CurrentCellDirtyStateChanged;
            }
            private void DataGridView1_CellValueChanged(object sender,
DataGridViewCellEventArgs e){
                if (AreAllCellsFilled(dataGridView1)){
                    richTextBox1.AppendText("Заповнено всі клітинки!\n");
                    button4.Enabled = true;
                }}
            private void DataGridView1_CurrentCellDirtyStateChanged(object
sender, EventArgs e){
                if (dataGridView1.IsCurrentCellDirty){
dataGridView1.CommitEdit(DataGridViewDataErrorContexts.Commit);
                }}
            private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e){
                if (e.ColumnIndex == -1 || e.RowIndex == -1)return;
                if (e.ColumnIndex == E){
                    comboBox2.Enabled = true;
                    button3.Enabled = true;
                }else{
                    comboBox2.Enabled = false;
                    button3.Enabled = false;
                }
                selectedCell =
dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex];
            }
            private void button3_Click(object sender, EventArgs e){
                selectedCell.Value = comboBox2.Text;
            }
            private void button5_Click(object sender, EventArgs e){
                int num_prior=int.Parse(dataGridView1.Rows.Count.ToString());
                int[] norm_p=new int[num_prior];
                richTextBox1.AppendText("norm_p = {");
                for (int i = 0; i < num_prior; i++){
                    norm_p[i] = num_prior - i;
                    richTextBox1.AppendText($"{norm_p[i]} ");
                }
                richTextBox1.AppendText("}\n");
                double[] real_prior = new double[num_prior];
                for (int i = 0; i < Q; i++){
                    var numValue = dataGridView1.Rows[i].Cells[E + 1].Value;
                    if (int.TryParse(numValue?.ToString(), out int number)){
                        richTextBox1.AppendText($"Приорітет: {number}\n");
                        for (int pr_index = 0; pr_index < num_prior;
pr_index++){
                   if (number == norm_p[pr_index]){ real_prior[i] = pr_index+1;
richTextBox1.AppendText($"Реальний приорітет: {real_prior[i]}\n");}
                        }}
                    else {
                       string pattern = @"^\d+-\d+$";
            if (numValue != null && Regex.IsMatch(numValue.ToString(), pattern))
```

```
                            {
                                richTextBox1.AppendText($"Формат 'число-число':
{numValue}\n");
                                string[] parts = numValue.ToString().Split('-');
           if (int.TryParse(parts[0], out int number1) && int.TryParse(parts[1],
out int number2))
                                {
 richTextBox1.AppendText($"Перше число: {number1}, друге число: {number2}\n");
                                    int r_num1 = 0;
                                    int r_num2 = 0;
                               for (int pr_index = 0; pr_index < num_prior; pr_index++){
                                if (number1 == norm_p[pr_index]){r_num1 = pr_index + 1;}
                                    }
                                    for (int pr_index = 0; pr_index < num_prior;
pr_index++)
                                    {
                                        if (number2 == norm_p[pr_index])
                                        {
                                            r_num2 = pr_index + 1;
                                        }
                                    }
       richTextBox1.AppendText($"Перше число: {r_num1}, друге число: {r_num2}\n");
                                    real_prior[i] = (r_num1*1.0+r_num2*1.0)/2;
          richTextBox1.AppendText($"Реальний приорітет: {real_prior[i]}\n");
                                }}}
                    }
                    richTextBox1.AppendText("real_p = {");
                    double sum_r_pr = 0;
                    for (int i = 0; i < num_prior; i++){
                        richTextBox1.AppendText($"{real_prior[i]} ");
                        sum_r_pr += real_prior[i];
                    }
                    richTextBox1.AppendText("}\n");
                    richTextBox1.AppendText($"Сума приорітетів = {sum_r_pr}\n");
              dataGridView1.Columns.Add($"{E + 3}", "Ваги (просте ранжування)");
                    for (int i = 0; i < Q; i++){
      dataGridView1.Rows[i].Cells[E + 3].Value =
(real_prior[i]/sum_r_pr).ToString();
                    }
         richTextBox1.AppendText("Глобальні оцінки методом простого ранжування\n");
                    double[] globe_arr = new double[E];
                    for (int i = 0; i < E; i++){
                        double sum_gl_E = 0;
                        for (int j = 0; j < Q; j++){
                            if
(double.TryParse(dataGridView1.Rows[j].Cells[i].Value.ToString(), out double
val)){ sum_gl_E += val * (real_prior[j] / sum_r_pr);}}
                        globe_arr[i] = sum_gl_E;
                        richTextBox1.AppendText($"E{i+1} = {globe_arr[i]}\n");
                    }
                    double sum_prop_k= 0;
                    for (int i = 0; i < Q; i++){
                        if
(double.TryParse(dataGridView1.Rows[i].Cells[E+2].Value.ToString(), out double
val)) { sum_prop_k += val;}}
              dataGridView1.Columns.Add($"{E + 4}", "Вага (пропорційний метод)");
                    double[] weight = new double[Q];
                    for (int i = 0; i < Q; i++){
if (double.TryParse(dataGridView1.Rows[i].Cells[E + 2].Value.ToString(), out
double val)){
                            weight[i]=val/sum_prop_k;
```

```
                        dataGridView1.Rows[i].Cells[E +
4].Value=weight[i].ToString(); }
                }
            richTextBox1.AppendText("Глобальні оцінки пропорційним методом\n");
                double[] globe_arr_prop = new double[E];
                for (int i = 0; i < E; i++){
                    double sum_gl_E = 0;
                    for (int j = 0; j < Q; j++){
                        if
(double.TryParse(dataGridView1.Rows[j].Cells[i].Value.ToString(), out double val))
                        {sum_gl_E += val * weight[j];}}
                    globe_arr_prop[i] = sum_gl_E;
                    richTextBox1.AppendText($"E{i + 1} =
{globe_arr_prop[i]}\n");
                }
            double maxValue = globe_arr[0];
                int maxIndex = 0;
                for (int i = 1; i < globe_arr.Length; i++){
                    if (globe_arr[i] > maxValue){
                        maxValue = globe_arr[i];
                        maxIndex = i;}}
                richTextBox1.AppendText($"Оптимальне рішення ( за методом
простого ранжування ) : E{maxIndex+1}\n");
                maxValue = globe_arr_prop[0];
                maxIndex = 0;
                for (int i = 1; i < globe_arr_prop.Length; i++){
                    if (globe_arr_prop[i] > maxValue){
                        maxValue = globe_arr_prop[i];
                        maxIndex = i;
                    }
                }
                richTextBox1.AppendText($"Оптимальне рішення ( за пропорційним
методом ) : E{maxIndex+1}\n");
              }
              private void button4_Click(object sender, EventArgs e){
                    for (int i = 0; i < Q; i++)
                    {
                        if (dataGridView1.Rows[i].Cells[E].Value != null &&
dataGridView1.Rows[i].Cells[E].Value.ToString() == "Minimum")
                        {
                            double number = 0;
                            for (int j = 0; j < E; j++)
                            {
                                var numValue = dataGridView1.Rows[i].Cells[j].Value;
                                if (double.TryParse(numValue.ToString(), out double
val)){
                                    if (val != 0){
                                        number = 1 / val;
      dataGridView1.Rows[i].Cells[j].Value = number.ToString();
                                    }
                                    else{
richTextBox1.AppendText($"Помилка: ділення на нуль у ({i + 1}, {j + 1})!\n");
                                    }}}}}
                    dataGridView1.Columns.Add($"{E + 1}", "Пріорітет");
                    dataGridView1.Columns.Add($"{E + 2}", "Коефіцієнт");
                    richTextBox1.AppendText("Нормалізацію проведено!\n");
                    button4.Enabled = false;
                    button5.Enabled = true;}}}
```

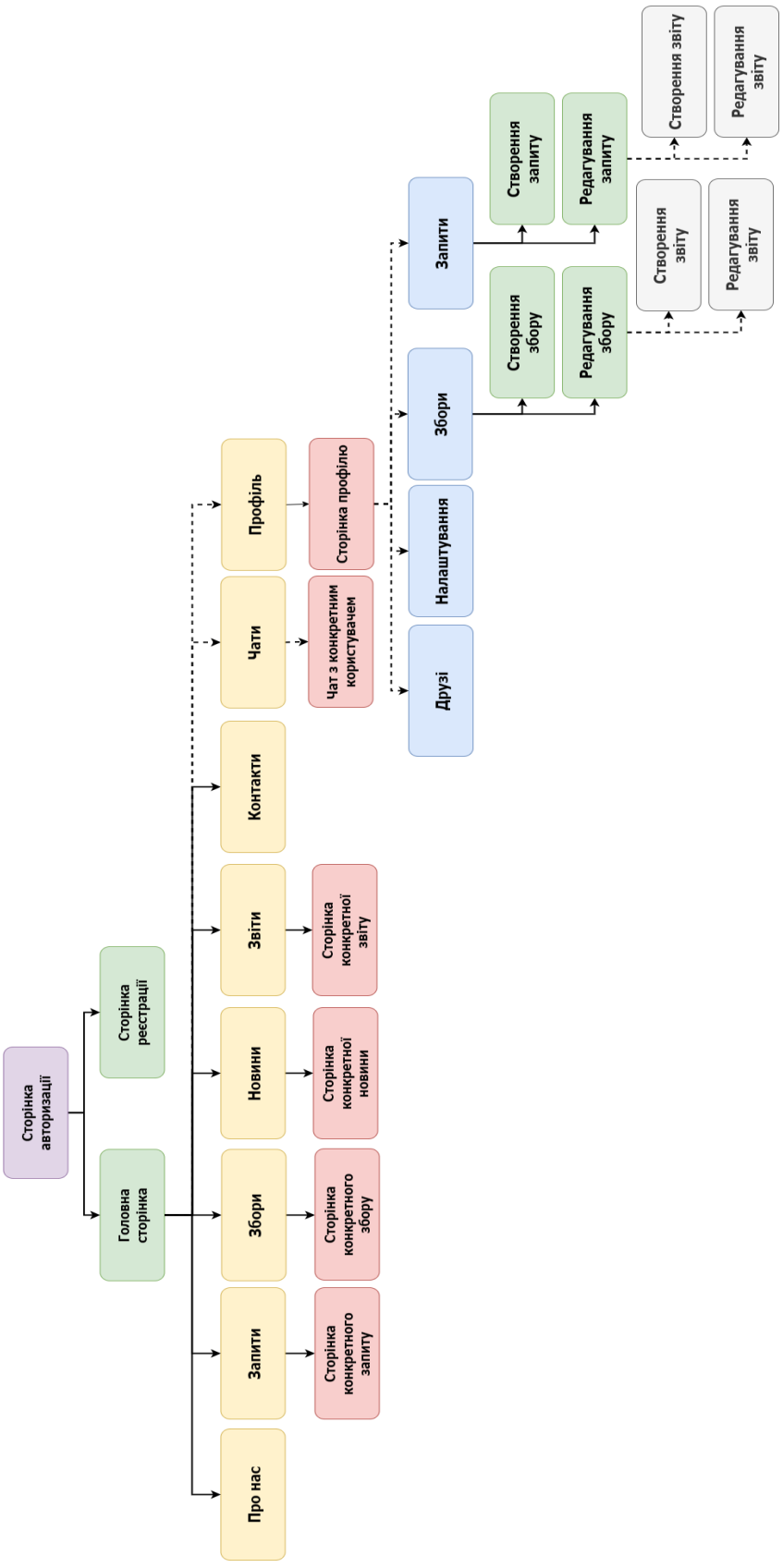
ДОДАТОК Б

Схема зв'язків БД



ДОДАТОК В

Карта вебзастосунку (Site Map)



ДОДАТОК Г

Програмний код вебзастосунку

```
const routes = [
        { path: "/", component: SignInPage },
        { path: "/registration", component: RegistrationPage },
        { path: "/home", component: Home },
        { path: "/about", component: AboutUsPage },
        { path: "/contacts", component: ContactPage },
        { path: "/news", component: NewsPage },
        …
        { path: "/change-req/:id", component: ChangeRequestPage },
      ];
      const router = createRouter({
        history: createWebHistory(),
        routes,
        scrollBehavior() {
          return { top: 0 };
        },
      });
      export default router
      //Основний файл стилів _main.scss
      @import
url("https://fonts.googleapis.com/css2?family=Lora:ital,wght@0,400..700;1,400..700
&display=swap");
      …
      $background-color1: #f5f5f5;
      $background-color2: #e4e5ee;
      $button-color: #ffd700;
      $title: #003da6;
      $subtitle: #005bbb;
      $main-text-color: #333333;
      $green-color: #02b72c;
      $orange-color: #ffe100;
      $red-color: #b71d02;

      %home-page-title-text {
        font-family: "Playfair Display", serif;
        font-optical-sizing: auto;
        font-weight: 700;
        font-style: normal;
        font-size: 56px;
        color: $main-text-color;
      }…
      // JS-логіка сторінки авторизації
        methods: {
          openSession() {
            fetch("http://localhost/heart_ua/sign_in.php", {
              method: "POST",
              headers: {
                "Content-Type": "application/json",
              },
              body: JSON.stringify({
                login: this.login,
                password: this.password,
              }),
            })
              .then((response) => response.json())
              .then((data) => {
```

```
                console.log("Success:", data);
                if (data["status"] == "success") {
                  this.createSession(data["id"]);
                  this.$router.push("/home");
                } else {
                  alert(data["message"]);
                }
              })
              .catch((error) => {
                console.error("Error:", error);
              });
          },
          createSession($id) {
            sessionStorage.setItem("id", $id);
          },
        },
      };
//PHP код обробки даних авторизації
      if ($_SERVER["REQUEST_METHOD"] == "POST") {
          $data = json_decode(file_get_contents("php://input"), true);
          $login = $data["login"];
          $pswd = $data["password"];
          try {
              $server = "localhost";
              $user = "root";
              $port = 3307;
              $password = "";
              $db = "HeartUA";
      $dbcon    =    new    PDO("mysql:host=$server;port=$port;dbname=$db",    $user,
$password);
              $dbcon->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
              $sql = "SELECT * FROM Users WHERE login = :login";
              $stmt = $dbcon->prepare($sql);
              $stmt->execute(['login' => $login]);
              $user = $stmt->fetch(PDO::FETCH_ASSOC);
              if ($user) {
                  if ($pswd == $user['password']) {
                      echo json_encode([
                          'status' => 'success','message' => "Вхід успішний",'id'
=> $user['id']]);
                  } else {
                      echo json_encode([
                          'status' => 'error',
                          'message' => "Невірний пароль",
                      ]);
                  }
              } else {
                  echo json_encode([
                      'status' => 'error','message' => "Невірний логін"]);
              }
          } catch (PDOException $e) {
              echo json_encode([
                  'status' => 'error','message' => $e->getMessage()]);
          }
//JS логіка сторінки реєстрації нових користувачів
        methods: {
          switchShowingPassword() {
            this.showPassword = !this.showPassword;
          },
          switchShowingCheckPassword() {
            this.showCheckPassword = !this.showCheckPassword;
```

```
        },
        isUser18OrOlder(birthdayStr) {
          const today = new Date();
          const birthDate = new Date(birthdayStr);
          const age = today.getFullYear() - birthDate.getFullYear();
          const m = today.getMonth() - birthDate.getMonth();
          if (m < 0 || (m === 0 && today.getDate() < birthDate.getDate())) {
            return age - 1 >= 18;
          }
          return age >= 18;
        },
        async isNewLogin() {
          try {
            const response = await fetch(
              "http://localhost/heart_ua/check_login.php",
              {
                method: "POST",
                headers: { "Content-Type": "application/json",},
                body: JSON.stringify({ login: this.login }),
              }
            );
            const data = await response.json();
            return data["status"] == "not_found";
          } catch (error) {
            console.error("Error:", error);
            return false;
          }
        },
        async isNewEmail() {
          try {
            const response = await fetch(
              "http://localhost/heart_ua/check_email.php",
              {
                method: "POST",
                headers: {
                  "Content-Type": "application/json",
                }, body: JSON.stringify({ email: this.email }),
              }
            );
            const data = await response.json();
            return data["status"] == "not_found";
          } catch (error) {
            console.error("Error:", error);
            return false;
          }
        },
        async goRegistration() {
          if (!this.isAgree) {
            alert("Надайте згоду на обробку персональних даних!");
            return;
          }
          if (
            this.login == "" || this.email == "" || this.phone == "" ||
this.password == "" || this.cpassword == "" || this.name == "" || this.lastName ==
"" || this.secondName == "" || this.birthday == "" ||this.passport == "") {
            alert("Введіть всі необхідні данні для реєстрації!");
            return;
          }
          const isLoginFree = await this.isNewLogin();
          const isEmailFree = await this.isNewEmail();
          if (!isLoginFree) {
```

```
            alert("Вибачте, але цей логін вже використовується!");
            return;
          }
          if (!isEmailFree) {
            alert("Вибачте, але на цю електронну пошту вже зареєстровано користувача!");
            return;
          }
          …
          let user = {
            login: this.login, email: this.email, phone: this.phone, password: this.password, cpassword: this.cpassword, name: this.name, lastName: this.lastName, secondName: this.secondName, birthday: this.birthday,passport: this.passport,
          };
          console.log(user);
          fetch("http://localhost/heart_ua/registration.php", {
            method: "POST",
            headers: {
              "Content-Type": "application/json",
            },
            body: JSON.stringify(user),
          })
            .then((response) => response.json())
            .then((data) => {
              console.log("Success:", data);
              alert("Дякуємо за реєстрацію!");
              this.$router.push("/");
            })
            .catch((error) => {
              console.error("Error:", error);
            });
        },
      },
    };
```

// PHP код додання нового користувача у БД

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $data = json_decode(file_get_contents("php://input"), true);
    $login = $data["login"];
    $email = $data["email"];
    …
    $dotenv = Dotenv\Dotenv::createImmutable(__DIR__);
    $dotenv->load();
    $base64Key = $_ENV['ENCRYPTION_KEY'];
    $key = base64_decode($base64Key);
    $encryptedData = encryptText($passport, $key);
    try {
        $server = "localhost";
        $user = "root";
        $port = 3307;
        $password = "";
        $db = "HeartUA";
        $dbcon = new PDO("mysql:host=$server;port=$port;dbname=$db", $user, $password);
        $dbcon->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        echo json_encode([
            'status' => 'error',
            'message' => $e->getMessage()
        ]);
```

```
            }
            try {
                $stmt = $dbcon->prepare("
                INSERT INTO users (login, email, password, phone,name, lname, sname,
birthday, passport, passportIV, passportTag, dateRegistration) VALUES (:login,
:email, :password, :phone,:name, :lname, :sname, :birthday, :passport, :passportIV,
:passportTag, :dateRegistration)");
                $today = date("Y-m-d");
                $stmt->execute([
                    ':login'    => $login,':email'    => $email,
                    ':password' => $pswd,':phone'    => $phone,
                    …
                ]);
                echo json_encode([
                    'status' => 'success',
                    'message' => 'Пользователь успешно добавлен',
                ]);
            } catch (PDOException $e) {
                echo json_encode([
                    'status' => 'error',
                    'message' => $e->getMessage()
                ]);}}
```

// Функція шифрування алгоритму AES-256-GCM

```
        function encryptText(string $text, string $key): array{
            $cipher = 'aes-256-gcm';
            $ivLength = openssl_cipher_iv_length($cipher);
            $iv = random_bytes($ivLength);
            $tag = '';
            $ciphertext = openssl_encrypt( $text,$cipher,$key,OPENSSL_RAW_DATA,$iv,
$tag,'',16);
            return [
                'ciphertext' => base64_encode($ciphertext),
                'iv' => base64_encode($iv),
                'tag' => base64_encode($tag),
            ];
        }
```

// Стилі спливаючого вікна меню профілю (анімація)

```
        .overlay {
          animation: showOverlay 0.2s linear forwards;
          .overlay-content1,
          .overlay-content2 {
            animation: showOverlayContent 0.5s linear forwards;
          }
          &.overlay-leave-active {
            animation: hideOverlay 0.3s linear forwards;
          }
        }
        @keyframes showOverlay {
          0% {opacity: 0;}
          100% {opacity: 1;}
        }
        @keyframes hideOverlay {
          0% {opacity: 1;}
          100% {opacity: 0;}
        }
        @keyframes showOverlayContent {
          0% {transform: translateX(100%);}
          100% {transform: translateX(0%);}
        }
```

// HTML документ спливаючаго вікна чатів

```
<transition name="chats-overlay">
  <div v-if="isVisible" class="overlay" @click="closeOverlay">
    <transition name="overlay-content">
      <div v-if="isVisible" class="overlay-inner" @click.stop>
        <div v-if="chats.length === 0" class="chats-content1">
                <img  class="none-mess"  src="/none-message.png"  alt="Немає
повідомлень"/>
          <p class="none-mess-p"> Тут поки що немає початих розмов чи сповіщень
від інших користувачів.
           </p>
        </div>
        <div v-else class="chats-content2">
          <TheChat
            v-for="c in chats" :key="c.id" :image="c.image"
            :name="c.name" :lname="c.lname"
            …
          />
        </div>
      </div>
    </transition>
  </div>
</transition>
```

// JS логіка чату з конкретним користувачем

```
import TheMessage from "./TheMessage/TheMessage.vue";
import { EventBus } from "../../event-bus/event-bus";
export default {
  watch: {
    isVisible() {
      if (this.isVisible) {this.scrollToBottom();}
    },
  },
  methods: {
    closeOverlay() {
      this.$emit("close");
    },
    async addMessage() {
      const idUser = Number(sessionStorage.getItem("id"));
      try {
        const res = await fetch(
`http://localhost/heart_ua/add_message.php?idU=${idUser}&idR=${this.idPenUse
r}&message=${this.message}`
        );
        const data = await res.json();
        await this.fetchChat();
        console.log(data);
        this.scrollToBottom();
        this.message = "";
        EventBus.emit("chats-updated");
        console.log("Додано нове повідомлення!");
      } catch (error) {
        console.log("Помилка додавання нового повідомлення у чат: " + error);
      }
    },
    async readNewMessages() {
      try {
 await fetch(
`http://localhost/heart_ua/read_messages.php?idU=${this.idUser}&idR=${this.idPenUs
er}`);
```

```
            await this.fetchChat();
            console.log("Ми прочитали всі непрочитані повідомлення!");
          } catch (error) {
            console.log("Помилка читання непрочитаних повідомлень " + error);
          }
        },
        onScroll() {
          const container = this.$refs.scrollContainer;
          this.isAtBottom =
            container.scrollHeight       -       container.scrollTop       ===
container.clientHeight;
          if (this.isAtBottom) {this.fetchChat();}
        },
        scrollToBottom() {
          this.$nextTick(() => {
            const container = this.$refs.scrollContainer;
            if (container) {container.scrollTop = container.scrollHeight;}
          });
        },
      },
// PHP код додання нового повідомлення у чаті з конкретним користувачем
      $stmt = $dbcon->prepare("INSERT INTO `Messages` (`idSendUser`, `idRecipient`,
`message`, `dateTime`, `status`) VALUES (:idU, :idR, :message, :dt, 'new')");
      $stmt->execute([
          ':idU' => $idU,':idR' => $idR,
          ':message' => $message,':dt' => $now
      ]);
      $lastId = $dbcon->lastInsertId();
      $stmt = $dbcon->prepare("SELECT id FROM Chats WHERE (idUser1 = :idU AND idUser2
= :idR) OR (idUser1 = :idR AND idUser2 = :idU)");
      $stmt->execute([':idU' => $idU,':idR' => $idR]);
      if ($stmt->rowCount() > 0) { $stmt = $dbcon->prepare("UPDATE Chats SET
lastMessage = :lastId WHERE (idUser1 = :idU AND idUser2 = :idR) OR (idUser1 = :idR
AND idUser2 = :idU)");
          $stmt->execute([
              ':lastId' => $lastId,':idU' => $idU,':idR' => $idR
          ]);
      } else {
          $stmt = $dbcon->prepare("INSERT INTO Chats (idUser1, idUser2, lastMessage)
VALUES (:idU, :idR, :lastId)");
          $stmt->execute([
              ':idU' => $idU,':idR' => $idR,':lastId' => $lastId
          ]);}

      //HTML структура сторінки зі зборами

          <div class="sort-block">
            <select id="sort" name="sort" v-model="sort">
              <option value="">Сортування</option>
              <option value="new">Нові</option>
              <option value="almost-done">Майже завершені</option>
              <option value="low-goal">З найменшою ціллю</option>
              <option value="high-goal">З найбільшою ціллю</option>
            </select>
          </div>
          <div class="categories-block">
            <select id="category" name="category" v-model="category">
              <option value="">Категорія</option>
              <option value="Тварини">Тварини</option>
              <option value="Медичні">Медичні донати</option>
              <option value="Військові донати">Військові донати</option>
```

```
            <option value="Гум. допомога">Гум. допомога</option>
          </select>
        </div>
      </div>
      <MonetaryFeesList
        :search="this.search"
        :sort="this.sort"
        :category="this.category"
      />

    //JS логіка компоненту списку карток зборів
      watch: {
        search: "filterChange",
        sort: "filterChange",
        category: "filterChange",
        $route() {
          this.currentPage = 1;
          this.monetaryFeesCards = [];
          this.hasMoreMF = true;
          this.fetchMonetaryFees();
        },
      },
      methods: {
        filterChange() {
          this.currentPage = 1;
          this.monetaryFeesCards = [];
          this.hasMoreMF = true;
          this.fetchMonetaryFees();
        },
        async fetchMonetaryFees() {
          this.loading = true;
          const searchMF = encodeURIComponent(this.search);
          console.log(this.search);
          console.log(this.sort);
          console.log(this.search);
          try {
            const res = await fetch(
               `http://localhost/heart_ua/get_monetary_fees.php?page=${this.curre
ntPage}&limit=${this.cardsPerPage}&search=${searchMF}&sort=${this.sort}&category=$
{this.category}`
            );
            const data = await res.json();
            console.log(data);
            const mappedData = data.map((d) => {
              const rest = d.goal - d.amount;
              const percent =
                d.goal > 0 ? Math.round((d.amount / d.goal) * 100) : 0;
              return {...d, rest,percent,};
            });
            this.monetaryFeesCards.push(...mappedData);
            console.log(this.monetaryFeesCards);
            if (data.length < this.cardsPerPage) { this.hasMoreMF = false;}
            console.log("Дані про збори отримано з серверу успішно!");
          } catch (error) {
            console.log("Помилка отримання списку зборів з серверу: " + error);
          } finally {this.loading = false;}
        },
        loadMoreMF() {
          this.currentPage++;
          this.fetchMonetaryFees();},
      },
```

```
  mounted() {this.fetchMonetaryFees();},
};
```

//Запит на сервер для отримання списку зборів

```
$sql = "SELECT * FROM MonetaryFees WHERE goal != amount";
if ($search != "") {
        if (preg_match('/^#(\d+)$/', $search, $matches)) {
                $ID = intval($matches[1]);
                $sql .= " AND id=$ID";
        } else { $search = '%' . $search . '%';
                $search = $dbcon->quote($search);
                $sql .= " AND title LIKE $search";}
}
if ($category != "") {
        $category = $dbcon->quote($category);
        $sql .= " AND type=$category";}
if ($sort != "") {
        switch ($sort) {
                case 'new':$sql .= " ORDER BY id DESC";break;
                case 'almost-done': $sql .= " ORDER BY ((amount/goal)*100)
DESC"; break;
                case 'low-goal':$sql .= " ORDER BY goal ASC";break;
                case 'high-goal':$sql .= " ORDER BY goal DESC";break;
                default: break;}}
$sql .= " LIMIT $limit OFFSET $offset";
$stmt = $dbcon->query($sql);
$dev = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

//JS логіка сторінки конкретного збору

```
  methods: {
    async goToPaymentSystem() {
      const idU=sessionStorage.getItem("id");
      try {
        const mode = 2;
        const idMF = this.$route.params.id;
        const res = await fetch(

`http://localhost/heart_ua/create_payment.php?money=${this.moneyDonate}&mode=${mod
e}&idMF=${idMF}&idU=${idU}`
        );
        const data = await res.json();
        console.log(data);
        console.log("URL:" + data.response.checkout_url);
        if (data && data.response && data.response.checkout_url) {
          window.location.href = data.response.checkout_url;
        } else {
          console.error("Помилка від сервера:", data.error || data);
          alert("Сталася помилка при створенні платежу.");
        }
      } catch (error) {
        console.log("Помилка здійснення платежу!", error);
      }
    },
  mounted() {
    let back = document.getElementById("body");
    back.style = "overflow-x: hidden; background-color: #E4E5EE";
    this.fetchTheMF();
  },
  computed: {
```

```
        formattedGoal() {
          if (this.mf.goal > 1000 && this.mf.goal < 1000000) {
            return (this.mf.goal / 1000).toFixed(0) + " тис.";
          } else if (this.mf.goal > 999999) {
            return (this.mf.goal / 1000000).toFixed(0) + " млн.";
          }
          return this.goal;
        },
        formattedRest() {
          if (this.rest > 1000 && this.rest < 1000000) {
            return (this.rest / 1000).toFixed(0) + " тис.";
          } else if (this.rest > 999999) {
            return (this.rest / 1000000).toFixed(0) + " млн.";
          }
          return this.rest;
        },
      },
    };
```

//PHP код для створення платежу

```
    if ($money != 0) {
        $order_desc = "";
        $response_url = "";
        if ($idMF > 0) {
            $dataMF = [
                'idMF' => $idMF,
            ];

            $ch = curl_init();
           curl_setopt($ch, CURLOPT_URL, 'http://localhost/heart_ua/get_mf.php');
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLOPT_POST, true);
            curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($dataMF));

            $response = curl_exec($ch);
            curl_close($ch);
            $result = json_decode($response, true);
               $order_desc = isset($result['title']) ? "Донат на збір '" .
$result['title'] . "'" : 'Донат через Fondy';
                                                    $response_url =
"http://localhost/heart_ua/success_payment.php?mode={$mode}&idMF={$idMF}&money={$m
oney}&idU={$idU}";
        } else {
            $order_desc = 'Донат через Fondy';
                                                    $response_url =
"http://localhost/heart_ua/success_payment.php/{$mode}";
        }
        $merchant_id = '1396424';
        $secret_key = 'test';
        $order_id = 'donate_' . time();
        $amount = $money * 100;
        $currency = 'UAH';
        $data = [
            'request' => [
                'merchant_id' => $merchant_id,
                'order_id' => $order_id,
                'amount' => $amount,
                'currency' => $currency,
                'order_desc' => $order_desc,
```

```
                'response_url' => $response_url,
            ]
        ];

                                $data['request']['server_callback_url']           =
'http://localhost/heart_ua/feedback_payment.php';

        $flat = $data['request'];
        ksort($flat);
        array_unshift($flat, $secret_key);
        $signature = sha1(implode('|', $flat));
        $data['request']['signature'] = $signature;

        $ch = curl_init('https://pay.fondy.eu/api/checkout/url/');
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));
        curl_setopt($ch, CURLOPT_HTTPHEADER, ['Content-Type: application/json']);
        $res = curl_exec($ch);
        curl_close($ch);
        echo $res;
    } else {
        echo json_encode(['error' => 'Помилка створення платежу']);
        http_response_code(404);
        exit;
    }

    //Код, що виконується підсля успішно пройденої оплати

    if ($id == 2) {
        $host = "localhost";
        $user = "root";
        $password = "";
        $db = "HeartUA";
        $port = 3307;

           $dbcon   =   new   PDO("mysql:host=$host;port=$port;dbname=$db",   $user,
$password);
        $dbcon->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $sql = "SELECT * FROM MonetaryFees WHERE id = :idMF";
        $stmt = $dbcon->prepare($sql);
        $stmt->bindParam(':idMF', $idMF, PDO::PARAM_INT);
        $stmt->execute();

        $dev = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($dev === false) {
            echo json_encode(['error' => 'Запис не знайдено']);
            http_response_code(404);
            exit;
        }
        $new_amount = $dev['amount'] + $money;
        if ($new_amount >= $dev['goal']) {
            $new_amount = $dev['goal'];
            $sql = "INSERT INTO Reports (idDonate) VALUES (:idMF)";
            $stmt = $dbcon->prepare($sql);
            $stmt->bindParam(':idMF', $idMF, PDO::PARAM_INT);
            $stmt->execute();
        }
        $sql = "UPDATE MonetaryFees SET amount=:newAmount WHERE id = :idMF";
        $stmt = $dbcon->prepare($sql);
        $stmt->bindParam(':idMF', $idMF, PDO::PARAM_INT);
```

```
        $stmt->bindParam(':newAmount', $new_amount, PDO::PARAM_STR);
        $stmt->execute();
    }
    if ($idU > 0) {
        $stmt = $dbcon->prepare("SELECT numDonates FROM Users WHERE id=:idU");
        $stmt->execute([':idU' => $idU]);
        $dev = $stmt->fetch(PDO::FETCH_ASSOC);
        $nD = $dev['numDonates'] + 1;
        $stmt = $dbcon->prepare("UPDATE Users SET numDonates =:nD WHERE id=:idU");
        $stmt->execute([':nD' => $nD, ':idU' => $idU]);
    }
    header("Location: http://localhost:8081/success-payment/$id");
    exit;
```

//JS логіка компоненту TheRequestPage, тобто сторінки окремого запиту

```
  methods: {
    async addPhysicalAssistance() {
      console.log("Фіз.допомога");
      const idU = sessionStorage.getItem("id");
      if (idU > 0) {
        if (idU == this.req.idUser) {
          alert("Неможливо записатися на свій ж запит");
          return;
        }
        try {
          await fetch(
    `http://localhost/heart_ua/add_requestHelp.php?idU=${idU}&mode=1&idR=${this.
req.id}`);
          alert("Дякуємо що записалися на фізичну допомогу!");
        } catch (error) { console.log("Помилка запису на фізичну допомогу: "
+ error);
        }
      } else { alert("Зареєструйтеся на платфоормі, щоб допомагати у
запитах!");
      }
    },
    async addMaterials() {
      …
    },
    async addAdvice() {
…
    },
    handleClick() {
      if (this.req.type === "Фіз.допомога") {
        this.addPhysicalAssistance();
      } else if (this.req.type === "Матеріали") {
        this.addMaterials();
      } else {
        this.addAdvice();
      }
    },
    openAnotherUser() {
      if (sessionStorage.getItem("id") == null) {
        alert(
          "Спочатку ввійдіть в обліковий запис або зареєструйтеся, щоб
подивитися профілі інших користувачів"
        );
        return;
      }
      if (this.req.idUser != sessionStorage.getItem("id")) {
```

```
            this.$router.push(`/user/${this.req.idUser}`);
          } else {
            if (this.req.idUser === sessionStorage.getItem("id")) {
              this.$router.push(`/profile`);
            }
          }
        },
        async fetchRequestInfo() {
          const idReq = this.$route.params.id;
          try {const res = await fetch(
      `http://localhost/heart_ua/get_request.php?idReq=${idReq}`
            );
            const data = await res.json();
            console.log(data);
            this.req = data;
            console.log(this.req);
            this.mf = Array.isArray(data) ? data[0] : data;
            this.rest = this.mf.goal - this.mf.amount;
            this.percent = Math.round(
              (parseFloat(this.mf.amount) / parseFloat(this.mf.goal)) * 100
            );
            console.log("Дані про запит отримано з серверу успішно!");
          } catch (error) {
            console.log("Помилка отримання інформації запиту з серверу: " +
error);}
        },
      },
      mounted() {
        let back = document.getElementById("body");
        back.style = "overflow-x: hidden; background-color: #E4E5EE";
        this.fetchRequestInfo();
      },
    };
```

//PHP код відправки листа на корпоративну пошту

```
        $mail = new PHPMailer();
        try {
            $mail->isSMTP();
            $mail->Host       = 'smtp.ukr.net';
            $mail->SMTPAuth   = true;
            $mail->Username   = 'heart_ua@ukr.net';
            $mail->Password   = 'rzbkbSTCL4jk02os';
            $mail->SMTPSecure = 'ssl';
            $mail->Port       = 465;
            $mail->setFrom('heart_ua@ukr.net', 'HeartUA');
            $mail->addAddress('heart_ua@ukr.net');
            $mail->isHTML(true);
            $mail->Subject = $title;
            $mail->Body = "
            <html>
            <head>
            <style>
                body {
                    background-color: #e4e5ee;
                    padding: 40px;
                }
                p {
                    background-color: #f5f5f5;
                    padding: 10px;
                    border-radius: 12px;
```

```
                }
                strong {color: #003da6;}
            </style>
            </head>
            <body>
                <h2>Нове повідомлення від $name</h2>
                <p><strong>Email:</strong> $email</p>
                <p><strong>Тема:</strong> $title</p>
                <p><strong>Повідомлення:</strong><br>$message</p>
            </body>
            </html>
            ";
            $mail->CharSet = 'UTF-8';
            $mail->send();
            echo 1;
        } catch (Exception $e) {
            echo 0;
        }
    }
```

//Налаштування профілю (JS код)

```
  methods: {
          async SaveNewInformation() {
            const phoneRegex = /^\+380\d{9}$/;
            const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            if (!phoneRegex.test(this.newPhone)) {
              alert(
                "Невірний формат номеру. Номер телефону має починатися з +380 та містити 9 цифр відповідно."
              );
              return;
            }
            if (!emailRegex.test(this.newEmail)) {
              alert("Некорректный формат електронної пошти.");
              return;
            }
            if (confirm("Вы впевнені, що хочете оновити особисті дані?")) {
              const userID = sessionStorage.getItem("id");
              try {
                await fetch(
              `http://localhost/heart_ua/update_userInfo.php?idU=${userID}&nOrg=${
                    this.newOrganization
                  }&nP=${encodeURIComponent(this.newPhone)}&nE=${this.newEmail}`
                );
                await this.fetchUserInfo();
                console.log("Дані користувача оновлено успішно!");
                alert("Дані успішно оновлено!");
              } catch (error) {
                console.log("Помилка оновлення інформації про користувача: " + error);}}},
          async deleteAccount() {
            const userId = sessionStorage.getItem("id");
            try {
              const response = await fetch(
                `http://localhost/heart_ua//delete_account.php?idU=${userId}`);
              const data = await response.json();
              if (data.success) {
                alert("Обліковий запис успішно видалено.");
                this.$router.push("/");
              } else {alert("Помилка: " + data.error);}
            } catch (error) {
```

```
                console.error("Помилка при видаленні облікового запису:", error);
                alert("Сталася помилка при з'єднанні з сервером.");
              }},},
//Видалення облікового запису
            $sql = "SELECT id FROM Users WHERE id = :idU";
            $stmt = $dbcon->prepare($sql);
            $stmt->bindParam(':idU', $idU, PDO::PARAM_INT);
            $stmt->execute();
            if ($stmt->rowCount() === 0) {
                echo json_encode(["success" => false,"error" => "Користувача з таким
ID не існує."]);
                exit;}
            $sql = "SELECT id, goal, amount FROM MonetaryFees WHERE idUser = :idU";
            $stmt = $dbcon->prepare($sql);
            $stmt->bindParam(':idU', $idU, PDO::PARAM_INT);
            $stmt->execute();
            $monetaryFees = $stmt->fetchAll(PDO::FETCH_ASSOC);
            foreach ($monetaryFees as $mf) {
                if ($mf['amount'] < $mf['goal']) {
                    echo  json_encode(["success"  =>  false,  "error"  =>  "Неможливо
видалити обліковий запис — у Вас є незавершені збори"]);
                    exit;
                } else {
              $sqlReport = "SELECT id, isActive FROM Reports WHERE idDonate = :idMF";
                    $stmtReport = $dbcon->prepare($sqlReport);
                    $stmtReport->bindParam(':idMF', $mf['id'], PDO::PARAM_INT);
                    $stmtReport->execute();
                    $report = $stmtReport->fetch(PDO::FETCH_ASSOC);
                    if ($report && $report['isActive'] == 0) {
                        echo json_encode(["success" => false, "error" => "Неможливо
видалити обліковий запис — у Вас є незаповнений звіт по збору"
                        ]); exit;
                    } else {
                        $sql = "DELETE FROM Reports WHERE id=:idRep";
                        $stmt = $dbcon->prepare($sql);
                        $stmt->bindParam(':idRep', $report['id'], PDO::PARAM_INT);
                        $stmt->execute();
                    }}}
            $sql = "SELECT id, type, amount, goal FROM Requests WHERE idUser = :idU";
            $stmt = $dbcon->prepare($sql);
            $stmt->bindParam(':idU', $idU, PDO::PARAM_INT);
            $stmt->execute();
            $requests = $stmt->fetchAll(PDO::FETCH_ASSOC);
            foreach ($requests as $req) {
                if ($req['amount'] != NULL && $req['goal'] != NULL && $req['type'] !=
"Порада") {
                    if ($req['amount'] < $req['goal']) { echo json_encode(["success"
=> false, "error" => "Неможливо видалити обліковий запис — у Вас є незавершені
запити"]);exit;
                    } else {
          $sqlReport = "SELECT id, isActive FROM Reports WHERE idRequest = :idR";
                        $stmtReport = $dbcon->prepare($sqlReport);
                        $stmtReport->bindParam(':idR', $req['id'], PDO::PARAM_INT);
                        $stmtReport->execute();
                        $report = $stmtReport->fetch(PDO::FETCH_ASSOC);
                        if ($report && $report['isActive'] == 0) {
                            echo  json_encode(["success"  =>  false,  "error"  =>
"Неможливо видалити обліковий запис — у Вас є незаповнений звіт по запиту"
                            ]);
                            exit;
                        } else {
```

```
                    $sql = "DELETE FROM Reports WHERE id=:idRep";
                    $stmt = $dbcon->prepare($sql);
$stmt->bindParam(':idRep', $report['id'], PDO::PARAM_INT);
                    $stmt->execute();
                }}}}
```

…

ДОДАТОК Д

Можливості користувача на платформі