МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційне та програмне забезпечення вебзастосунку благодійного фонду»

здобувача освіти групи ІН-01 Топорова Андрія Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Андрій ТОПОРОВ

(підпис)

Керівник,
асистент кафедри комп'ютерних наук,
кандидат фізико-математичних наук Ольга ШУТИЛЄВА

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-01 Топорова Андрія Сергійовича

1. Тема роботи: «Інформаційне та програмне забезпечення вебзастосунку благодійного фонду»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 01 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Проєктування інформаційної моделі та моделі бази даних. 3) Програмна реалізація прототипу вебзастосунку благодійного фонду. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «08» квітня 2024 р.
Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	06.05.24-10.05.24	
2	*Проєктування інформаційної моделі та моделі бази даних.*	11.05.24-16.05.24	
3	*Розробка прототипу вебзастосунку благодійного фонду*	17.05.24-24.05.24	
4	*Аналіз отриманих результатів*	25.05.24-27.05.24	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	28.05.24-31.05.24	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 118 ст., 66 рис., 3 табл., 2 додатки, 31 використане джерело.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі функціонування роботи благодійного фонду шляхом розробки відповідного вебзастосунку.

Об'єкт дослідження — робота благодійного фонду.

Мета роботи — розробка інформаційного та програмного забезпечення вебзастосунку благодійного фонду.

Методи дослідження — способи проєктування та програмної реалізації вебзастосунків, та інструменти побудови інформаційних моделей.

Результати — розроблено прототип клієнтської та серверної частини вебзастосунку благодійного фонду, адаптований під екрани мобільних пристроїв, яка надає змогу користувачу можливість реєструватися, приховати чутливі дані задля власної безпеки, підтверджувати власні дані, авторизуватися, змінювати пароль та інші дані, видаляти користувача, та додавати оголошення про надання допомоги. Проведено аналіз отриманих результатів.

ВЕБЗАСТОСУНОК, БЛАГОДІЙНИЙ ФОНД, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, POSTGRESQL, EXPRESS.JS, REACT, NODE.JS.

ЗМІСТ

ВСТУП

Після того як в Україні розпочалося повномасштабне вторгнення, відбулася друга після 2014 року хвиля підйому волонтерського руху. Згідно з даним джерелом [1], у перші місяці громадяни розпочали волонтерську діяльність задля опору російській агресії, а також допомоги постраждалим внаслідок бойових дій, обстрілів чи тимчасової окупації.

Автори [1] зазначають, що нині волонтерські організації здебільшого спеціалізуються і компенсують неповну інституційну спроможність держави у сферах: матеріального й технічного забезпечення бійців на фронті; надання медичної допомоги пораненим та постраждалим; матеріального забезпечення та психологічної допомоги внутрішньо переміщеним особам у тилу.

Актуальність. Згідно з опитуванням, проведеним компанією Gradus Research, результати якого опубліковані в липні 2023 року та де взяло участь 1097 респондентів [2], удвічі збільшилася кількість громадян, які допомагають шляхом волонтерства, захисної діяльності та допомоги фронту: з 22% у липні 2022 року до 43% у липні 2023 року. Також у цьому джерелі сказано, що 53% респондентів долучаються до допомоги самостійно.

Також компанія Gradus Research провела наступне опитування, в якому залучила 1212 респондентів, і оприлюднила результат дослідження в лютому 2024 року [3], згідно з яким кількість тих, хто регулярно бере участь у благодійних чи волонтерських зборах, збільшилася з 13% у жовтні 2022 році до 17% у лютому 2024 році, а також на 4% збільшилася кількість тих, хто бере участь час від часу: з 40% у жовтні 2022 році до 44% у лютому 2024 році.

Об’єкт дослідження. Спираючись на отримані джерела, створення вебзастосунку для підтримки діяльності волонтерської організації досі залишається актуальним. Серед переваг виділяють основні з них:

1. Багато користувачів можуть мати доступ до однієї і тієї ж версії застосунку.

2. Не потребують встановлення.

3. Є кросплатформенними, тобто можна отримати доступ як зі стаціонарних комп’ютерів, так і з ноутбуків, планшетів, смартфонів.

4. Користувачі можуть отримати доступ до вебдодатку з різноманітних браузерів [4].

Таким чином, вебдодатки зможуть охопити ширшу авдиторію, а отже збільшиться кількість людей, які зацікавляться волонтерською діяльністю та надихнуть інших долучитися, і з цього випливає, що збільшиться як допомога Силам оборони України, так і постраждалим від наслідків війни.

Гіпотеза. Створення прототипу динамічного вебзастосунку діяльності благодійного фонду, що надаватиме можливості для надання допомоги як Силам оборони України, так і для надання гуманітарної допомоги.

Новизна. Даний прототип надаватиме можливість тим, хто хоче приховати чутливу інформацію задля власної безпеки, додавати оголошення про надання допомоги чи зборів.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасний стан

Як відомо, після 24 лютого 2022 року, коли розпочалося повномасштабне вторгнення, відбулася друга після 2014 року хвиля підйому волонтерського руху в Україні. Згідно з даним джерелом [1], у перші місяці громадяни розпочали волонтерську діяльність задля опору російській агресії, а також допомоги постраждалим внаслідок бойових дій, обстрілів чи тимчасової окупації.

Автори [1] зазаначають, що нині волонтерські організації здебільшого спеціалізуються і компенсують неповну інституційну спроможність держави у сферах: матеріального й технічного забезпечення бійців на фронті; надання медичної допомоги пораненим та постраждалим; матеріального забезпечення та психологічної допомоги внутрішньо переміщеним особам у тилу.

Згідно з опитуванням, проведеним компанією Gradus Research, результати якого опубліковані в липні 2023 року [2], удвічі збільшилася кількість громадян, які допомагають шляхом волонтерства, захисної діяльності та допомоги фронту: з 22% у липні 2022 року до 43% у липні 2023 року. Також у цьому джерелі сказано, що 53% респондентів долучаються до допомоги самостійно.

Також компанія Gradus Research провела наступне опитування, в якому залучила 1212 респондентів, і оприлюднила результат дослідження в лютому 2024 року [3], згідно з яким кількість тих, хто регулярно бере участь у благодійних чи волонтерських зборах, збільшилася з 13% у жовтні 2022 році до 17% у лютому 2024 році, а також на 4% збільшилася кількість тих, хто бере участь час від часу: з 40% у жовтні 2022 році до 44% у лютому 2024 році.

Спираючись на отримані джерела, створення вебзастосунку для підтримки діяльності волонтерської організації досі залишається актуальним. Серед переваг виділяють основні з них:

1. Багато користувачів можуть мати доступ до однієї і тієї ж версії застосунку.

2. Не потребують встановлення.

3. Є кросплатформенними, тобто можна отримати доступ як зі стаціонарних комп’ютерів, так і з ноутбуків, планшетів, смартфонів.

4. Користувачі можуть отримати доступ до вебдодатку з різноманітних браузерів [4].

Таким чином, вебдодатки зможуть охопити ширшу авдиторію, а отже збільшиться кількість людей, які зацікавляться волонтерською діяльністю та надихнуть інших долучитися, і з цього випливає, що збільшиться як допомога Силам оборони України, так і постраждалим від наслідків війни.

1.2 Аналіз аналогічних проєктів

Перед розробкою прототипу вебдодатку варто провести пошук та аналіз аналогів існуючих волонтерських організацій.

Аналіз проводитиметься за допомогою наступних 6 критеріїв:

1. Вміст сайту (content).
2. Дизайн (design).
3. Безпека (security).
4. Швидкодія (speed).
5. Зручність використання (usability).
6. Якість.

Перший аналог – вебдодаток благодійного фонду «Повернись живим» [5]. Дана організація спеціалізується на допомозі Силам оборони України та учасникам бойових дій. На Рисунок 1.1-Рисунок 1.4 наведено головна та інформаційна сторінки сайту, а також їхній вигляд на мобільних пристроях. Кольорова гама сайту підібрана у військовому стилі, що і підкреслює напрям діяльності БФ, пов’язаний з допомогою військовим. Також між кожними частинами сайту є проміжки, що дозволяють відділити кожен розділ додатку.

Використано шрифти Helvetica Neue, Helvetica та Laqonic з різними розмірами, які гармонійно поєднані між собою. У додатку наявна навігація, яка дозволяє перейти з однієї сторінки на іншу, є можливість докладніше дізнатися про організацію, наявні збори та звіти по зборам, дізнатися новини і контакти організації. На сайті є можливість завантажити річний звіт фонду у вигляді pdf-документу. Також вебдодаток є адаптований для мобільних пристроїв, що є однією з переваг, яка дозволяє охопити ще більшу авдиторію. Серед недоліків можна виділити низьку якість HTML-коду: за допомогою ресурсу validator.w3.org на головній сторінці виявлено 35 помилок та 86 попереджень на головній.

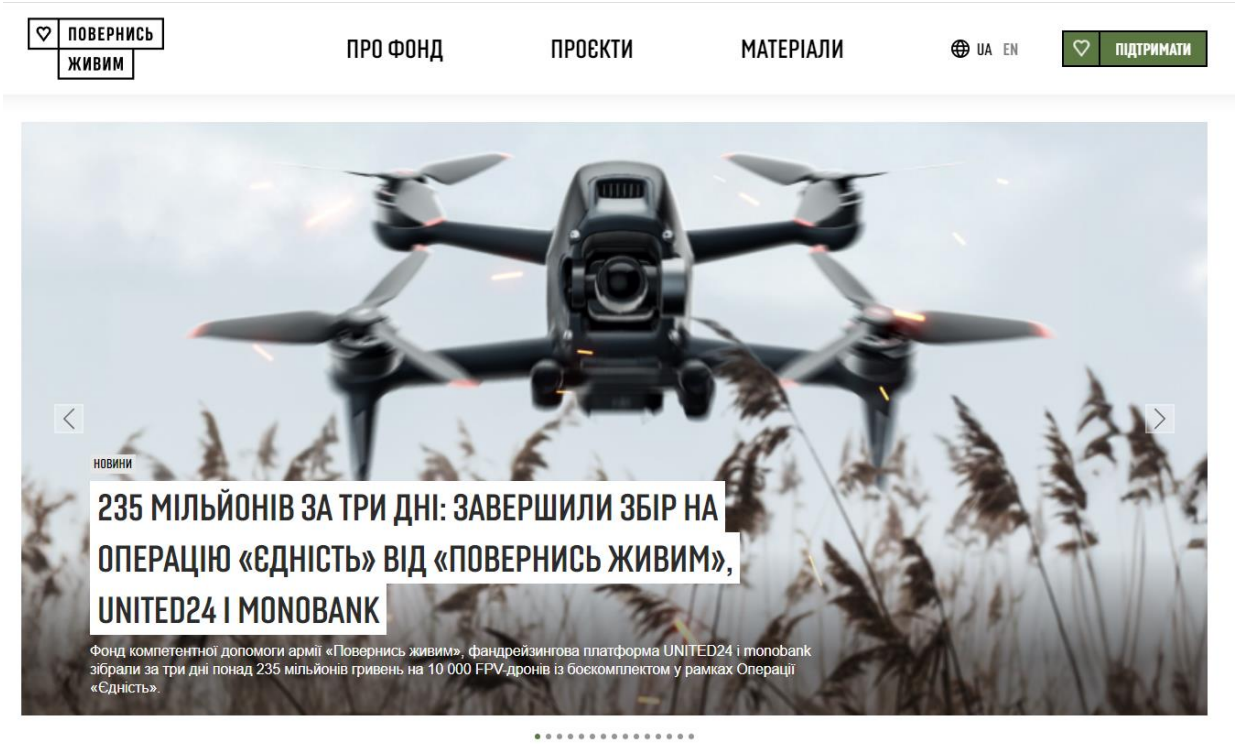

Рисунок 1.1 – Головна сторінка вебдодатку savelife.in.ua.

Рисунок 1.2 – Головна сторінка вебдодатку savelife.in.ua на мобільних пристроях.

Рисунок 1.3 – Інформаційна сторінка вебдодатку savelife.in.ua.

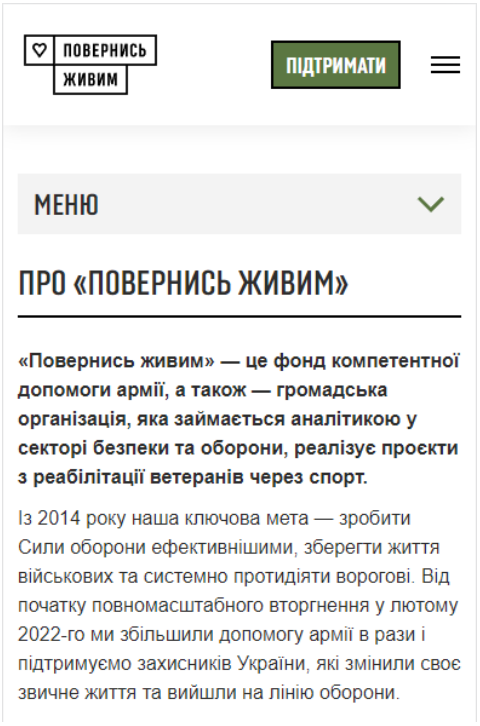

Рисунок 1.4 – Інформаційна сторінка вебдодатку savelife.in.ua на мобільних пристроях.

Другий аналог – вебзастосунок благодійного фонду Сергія Притули [6]. Організація спеціалізується як на допомозі Силам оборони, так і наданні гуманітарної допомоги. На Рисунок 1.5-Рисунок 1.8 наведено головна та інформаційна сторінки сайту, а також їхній вигляд на мобільних пристроях. Дизайн сайту нагадує дизайн вебпорталу державних послуг «Дія» [9]. Використано шрифт e-Ukraine-Medium з різними розмірами для розділів навігації, кнопок, заголовків розділів та тексту, і шрифти Roboto, RobotoDraft, Helvetica, Arial для таблиці звітності. У додатку можна дізнатися про діяльність благодійного фонду, новини фонду, збори та звіти, є контакти, а також розділ часто заданих питань, на які надано відповіді, наявний навігаційний блок у стелі (header) та підлозі (footer) сайту. Як і перший аналог, вебдодаток адаптований під мобільні пристрої. За допомогою ресурсу gtmetrix.com з’ясовано, що в порівнянні з попереднім аналогом, додаток має вищу оцінку продуктивності: 93% проти 60% у сайту фонду «Повернись живим», але гірша оцінка структури: 71% проти 76%. Серед недоліків можна

виділити недоліки в HTML-коді: нараховано 50 помилок на головній сторінці.

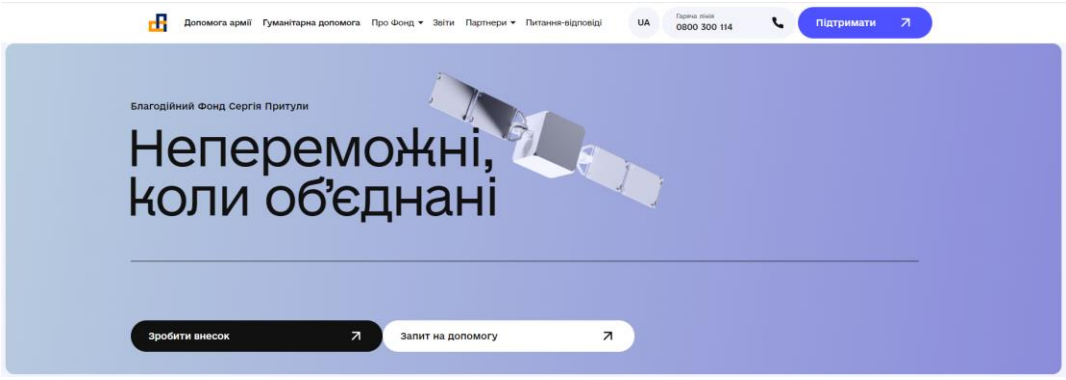

Рисунок 1.5 – Головна сторінка вебдодатку prytulafoundation.org.

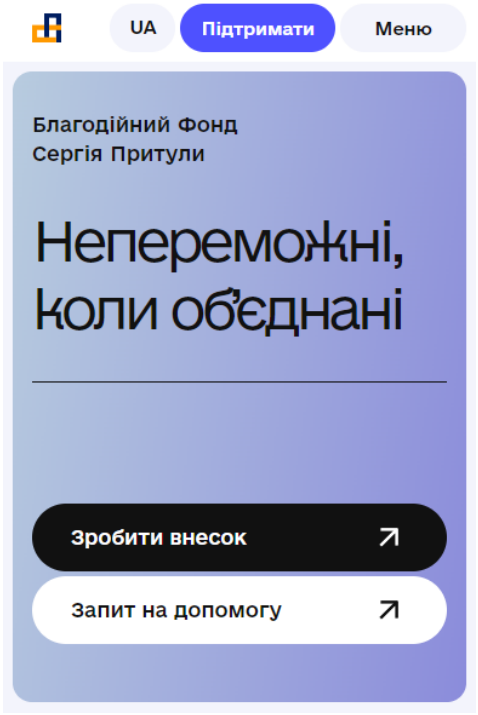

Рисунок 1.6 – Головна сторінка вебдодатку prytulafoundation.org на мобільних пристроях.

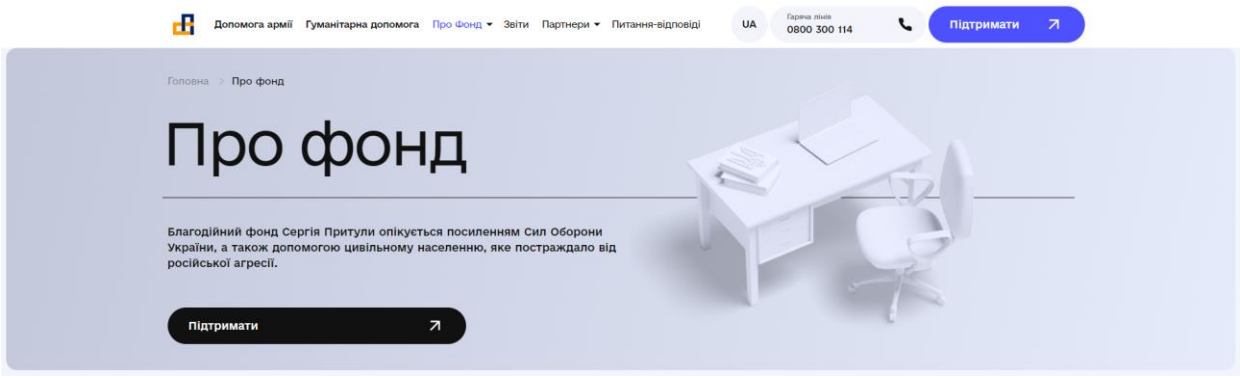

Рисунок 1.7 – Інформаційна сторінка вебдодатку prytulafoundation.org.

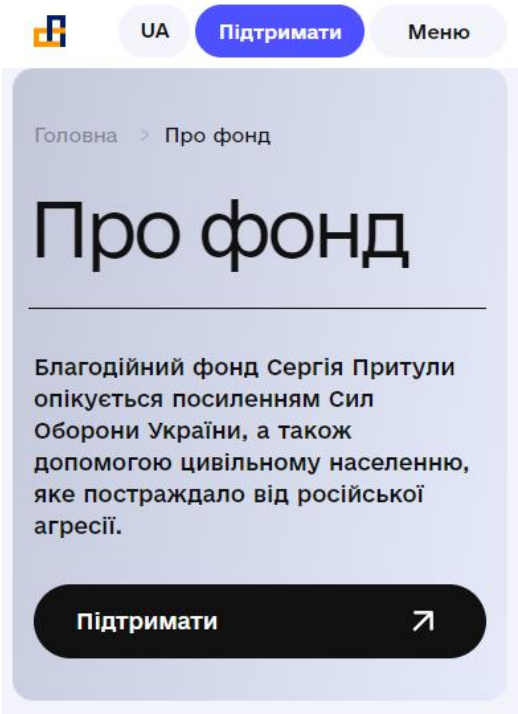

Рисунок 1.8 – Інформаційна сторінка вебдодатку prytulafoundation.org на мобільних пристроях.

Третій аналог – вебзастосунок благодійного фонду «Благомай» [7]. Основним напрямком благодійної діяльності фонду є допомога дітям з дитбудинків та їхній соціальній адаптації до реального життя. На Рисунок 1.9-Рисунок 1.12 наведено головна та інформаційна сторінки сайту, а також їхній вигляд на мобільних пристроях. У додатку використані шрифти Open Sans з різними розмірами та стилями. Як і в попередніх аналогах, сайт складається з головної, інформаційної сторінок, сторінок зборів, звітів та новин організації,

є контакти благодійного фонду. Дизайн, як і в попередніх сайтах, є адаптивним. З особливостей перед аналогами є наявність форми авторизації та реєстрації, а разом зі зборами відображається мапа, яка дає можливість побачити місцезнаходження установ, яким надається гуманітарна допомога. Із недоліків можна виділити найгірші оцінки продуктивності та структури: 44% та 46% проти 60% та 76% у сайту фонду «Повернись живим» і 93% та 71% у сайту фонду Сергія Притули, а також 27 помилок і 2 попередження в HTML-коді головної.

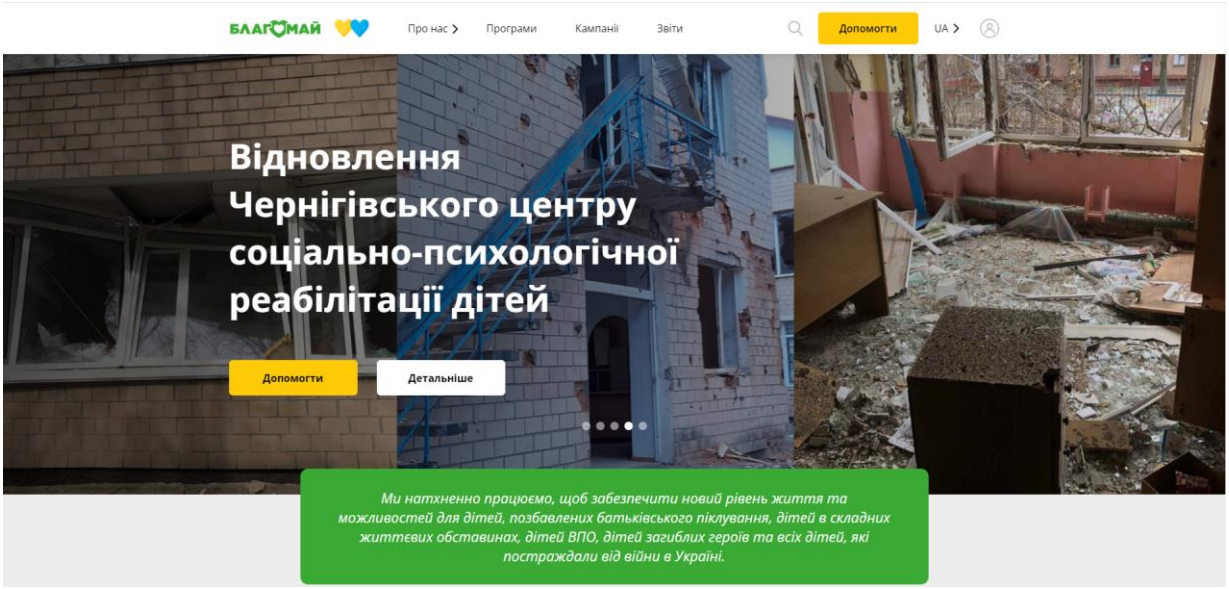

Рисунок 1.9 – Головна сторінка вебдодатку charitymay.com.

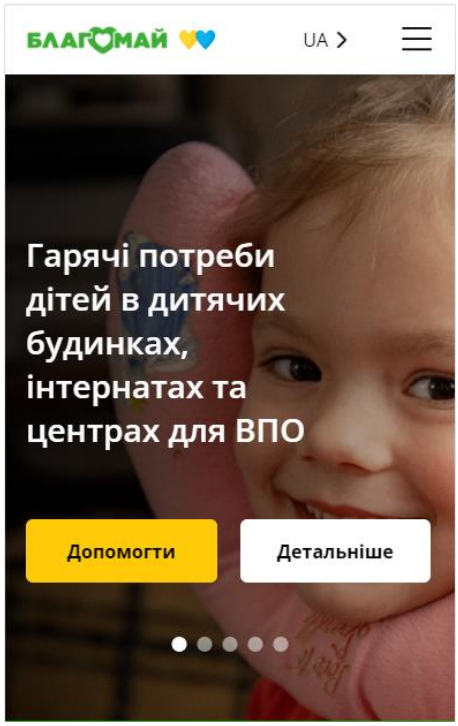

Рисунок 1.10 – Головна сторінка вебдодатку charitymay.com на мобільних пристроях.

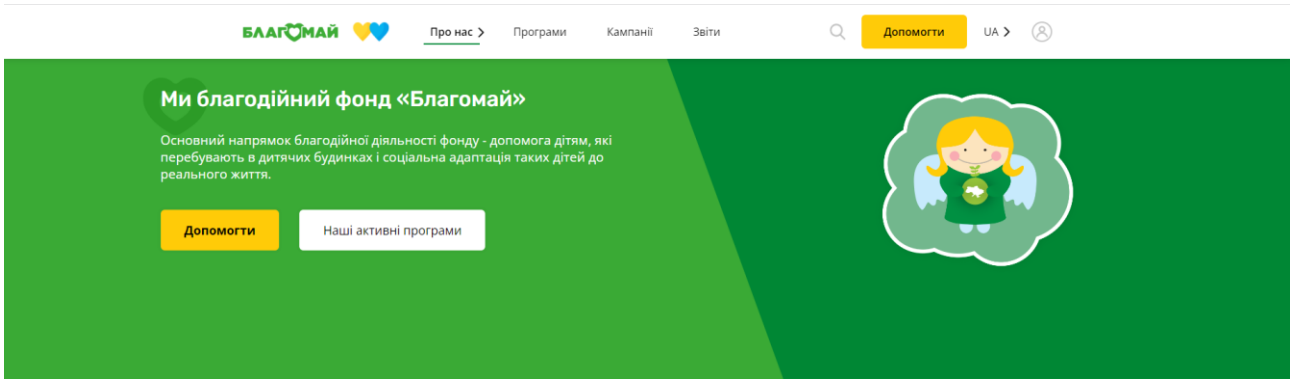

Рисунок 1.11 – Інформаційна сторінка вебдодатку charitymay.com.

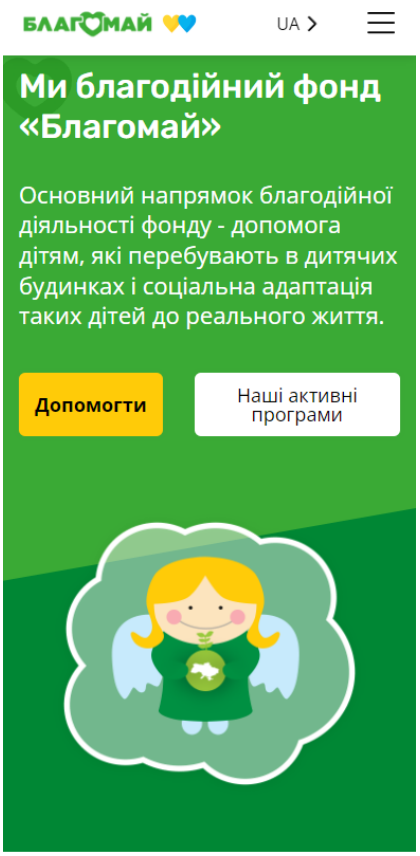

Рисунок 1.12 – Інформаційна сторінка вебдодатку charitymay.com на мобільних пристроях.

Четвертий аналог – вебдодаток благодійного фонду «Захист майбутнього» [8]. Даний фонд, в основному, надає допомогу військовим-добровольцям і військовим Сил Спеціальних Операцій ЗСУ. Вигляд головної та інформаційної сторінок можна побачити на Рисунок 1.13-Рисунок 1.16. У вебдодатку використано шрифт Rubik з використанням різних розмірів та стилів. Дизайн сайту зроблений у військовому стилі, як і дизайн сайту БФ «Повернись живим». Як і попередні аналоги, вебзастосунок містить навігаційне меню, головну, інформаційну сторінки, сторінки зборів, звітів, новин фонду. Як і попередні аналоги, дизайн сайту є адаптивним. Також даний вебдодаток показав одні з найвищих оцінок продуктивності та структури серед 4 аналогів: 90% та 86% відповідно, а також найменшу суму помилок і попереджень у HTML-коді: 15 помилок і 6 попереджень. У порівнянні з 4 аналогами, сайт є мінімалістичним і не містить нічого зайвого. До недоліків

можна віднести, як було сказано в одному з попередніх тверджень, наявність недоліків у HTML-коді.

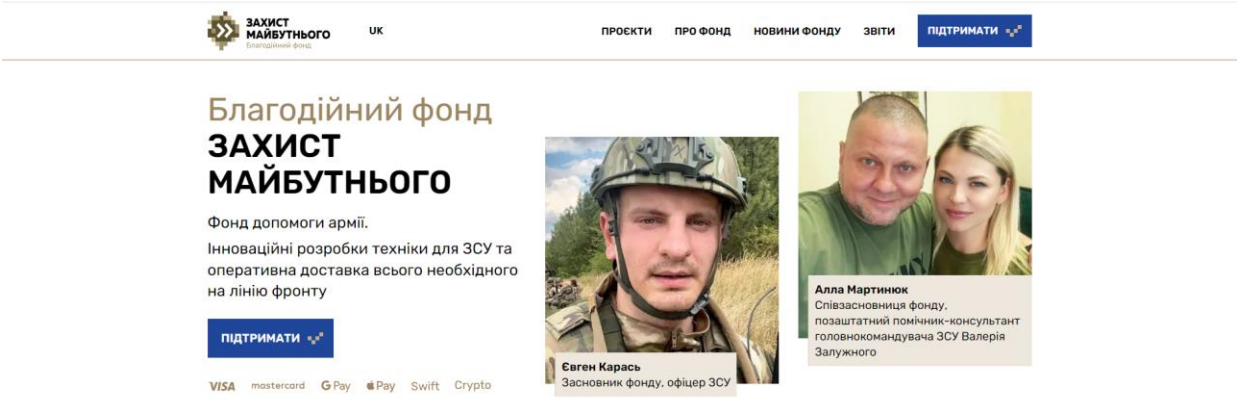

Рисунок 1.13 – Головна сторінка вебдодатку maibutniefund.org.

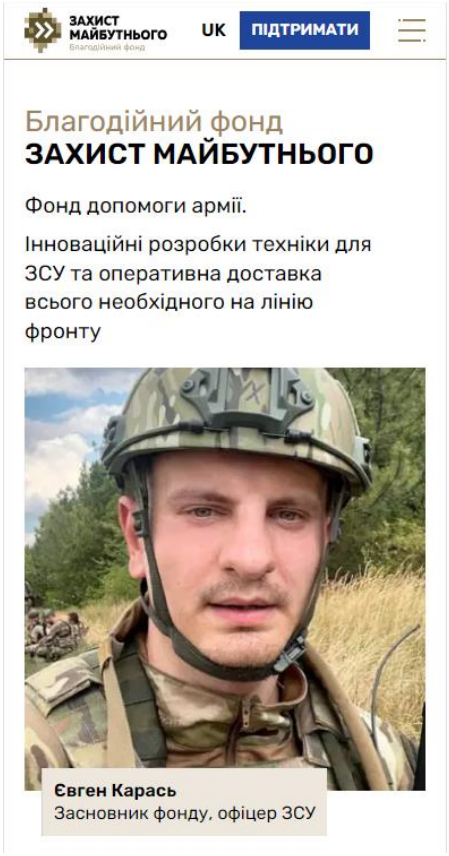

Рисунок 1.14 – Головна сторінка вебдодатку maibutniefund.org на мобільних пристроях.

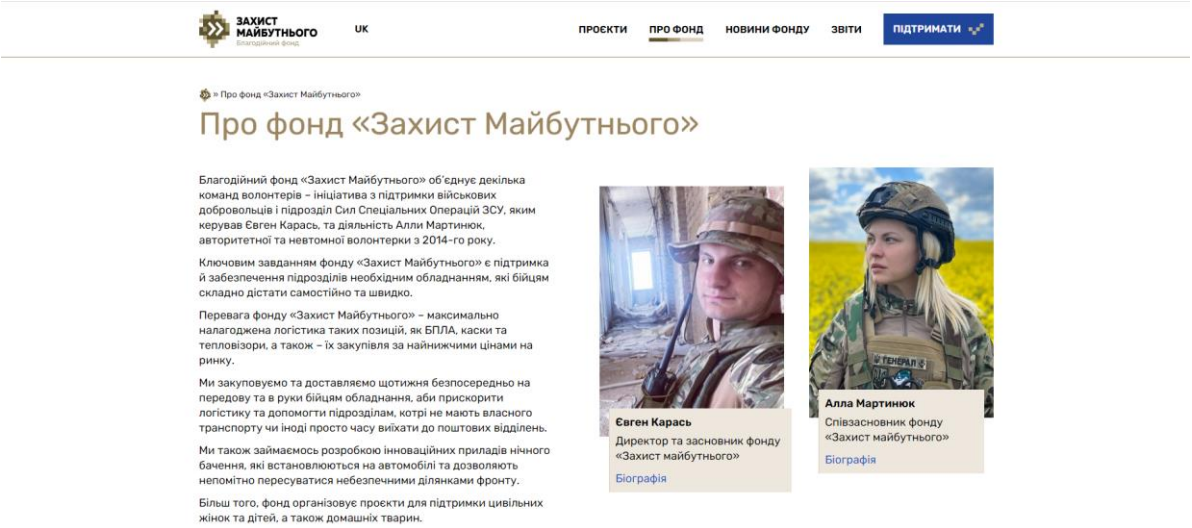

Рисунок 1.15 – Інформаційна сторінка вебдодатку maibutniefund.org.

Рисунок 1.16 – Інформаційна сторінка вебдодатку maibutniefund.org на мобільних пристроях.

Після детального аналізу 4 аналогів вебдодатків благодійних фондів, визначено їхні сильні та слабкі сторони, результати яких можна побачити на Таблиця 0.1.

Таблиця 0.1 – Порівняльна характеристика аналогів вебзастосунків

Критерії оцінювання\Веб-ресурс	**savelife.in.ua**	**prytulafoundation.org**	**charitymay.com**	**maibutniefund.org**
Зручний інтерфейс	+	+	+	+
Адаптивний дизайн	+	+	+	+
Інтерактивність	+	-	+	-
Головна	+	+	+	+
Інформаційна сторінка	+	+	+	+
Форма авторизації/реєстрації	-	-	+	-
Відсутність помилок/попереджень у HTML-коді	-	-	-	-
Відповідність контенту сайту до діяльності БФ	+	+	+	+
Надання допомоги Силам оборони	+	+	-	+
Надання гуманітарної допомоги	-	+	+	-
Оцінка продуктивності (у %)	60%	93%	44%	90%
Оцінка структури (у %)	76%	71%	46%	86%
Чи безпечне підключення до веб-додатку?	+	+	+	+

Після проведення аналізу додатків-аналогів, підводимо наступні підсумки. Кожен з них містить такі необхідні елементи, як навігаційне меню, головна, інформаційна сторінки, сторінки зборів, звітів, новин, контактні дані, а дизайн адаптований під мобільні девайси. Втім, щоб отримати якісний вебдодаток, який приваблюватиме користувачів, необхідно продумати його дизайн. З аналогів можна запозичити інформаційну складову, структуру,

ілюстративний матеріал та адаптивність під мобільні пристрої, при цьому додавши до веб-додатку нові функції, цим самим зробить програмний продукт особливим, привабливим для користувачів і дозволить усунути вищезазначені недоліки. Також серед усього цього варто додати можливість для користувачів приховати чутливу інформацію для решти користувачів, так що змогу побачити повну інформацію про користувача зможе лише адміністратор.

1.3 Постановка задачі

Метою даної роботи є створення прототипу вебзастосунку підтримки діяльності благодійного фонду. Його застосування дозволить забезпечити організацію БФ шляхом автоматизації взаємодії волонтерів чи волонтерських організацій, та людей, установ або організацій, яким необхідна допомога, таким чином долучивши велику кількість небайдужих до підтримки Сил оборони і постраждалим від наслідків війни.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) визначити актуальність розробки вебдодатку діяльності БФ та його цільову авдиторію;
2) виконати аналіз уже існуючих відповідно до тематики програмних продуктів і визначити їхні сильні та слабкі сторони;
3) обрати методи та технології розробки вебдодатку діяльності БФ;
4) спроєктувати модель та структуру створюваного вебдодатку;
5) створити прототип клієнтської та серверної частин вебдодатку;
6) провести аналіз отриманих результатів.

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Функціональна модель

Для функціонального моделювання та графічного опису процесів використано методологію IDEF0 [10]. Вона передбачає побудову ієрархічної системи діаграм – одиничних описів фрагментів системи.

Згідно зі стандартом IDEF0, діаграма складається з блоків, які називаються роботами, та стрілок. Роботи відповідають за функції, які виконуються протягом певного часу і мають результати. Стрілки позначають взаємодію робіт із зовнішнім світом і розділяються на 5 видів.

- Вхід – це об'єкти, що використовуються роботою для отримання виходу.
- Управління – це інформація, яка керує діями роботи.
- Вихід – це об'єкти, які є результатами роботи.
- Механізми – це ресурси, які виконують роботу.
- Виклики – це спеціальний вид стрілок, які позначають, що певна робота виконується за межами модельованої системи.

У контекстній діаграмі (Рисунок 2.1) визначені такі дані:

- Робота: організація благодійної діяльності.
- Вхід: запит на розміщення оголошення про збір.
- Управління: правила реєстрації/авторизації користувачів, інструкція з подання запиту, положення про доброчесність, адміністратор.
- Вихід: відмова в запиті, розміщене оголошення про збір.
- Механізми: вебдодаток БФ, апаратне забезпечення, сервер БД PostgreSQL.

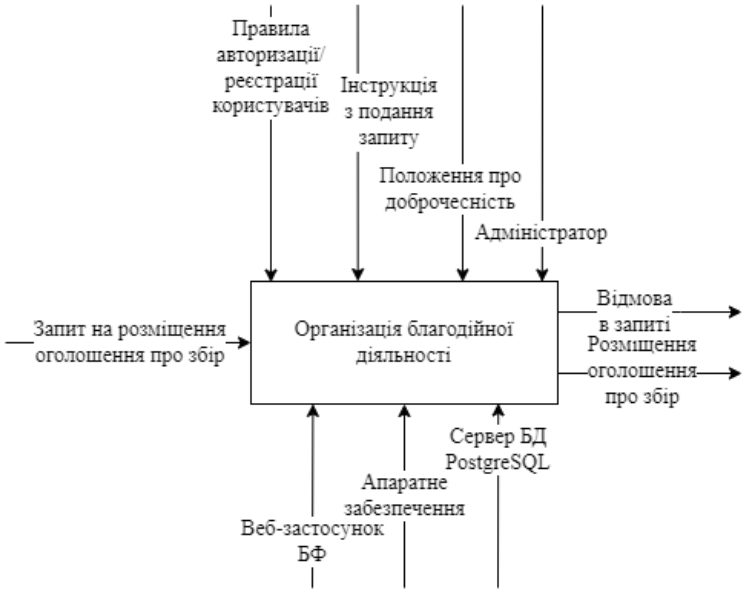

Рисунок 2.1 – Контекстна діаграма.

Наступним кроком проведено декомпозицію контекстної діаграми (Рисунок 2.2) для уточнення процесу шляхом розбиття одного процесу на декілька підпроцесів. У Таблиця 0.1 детально описано кожен підпроцес та які входи, виходи, управління та механізми використані при виконанні.

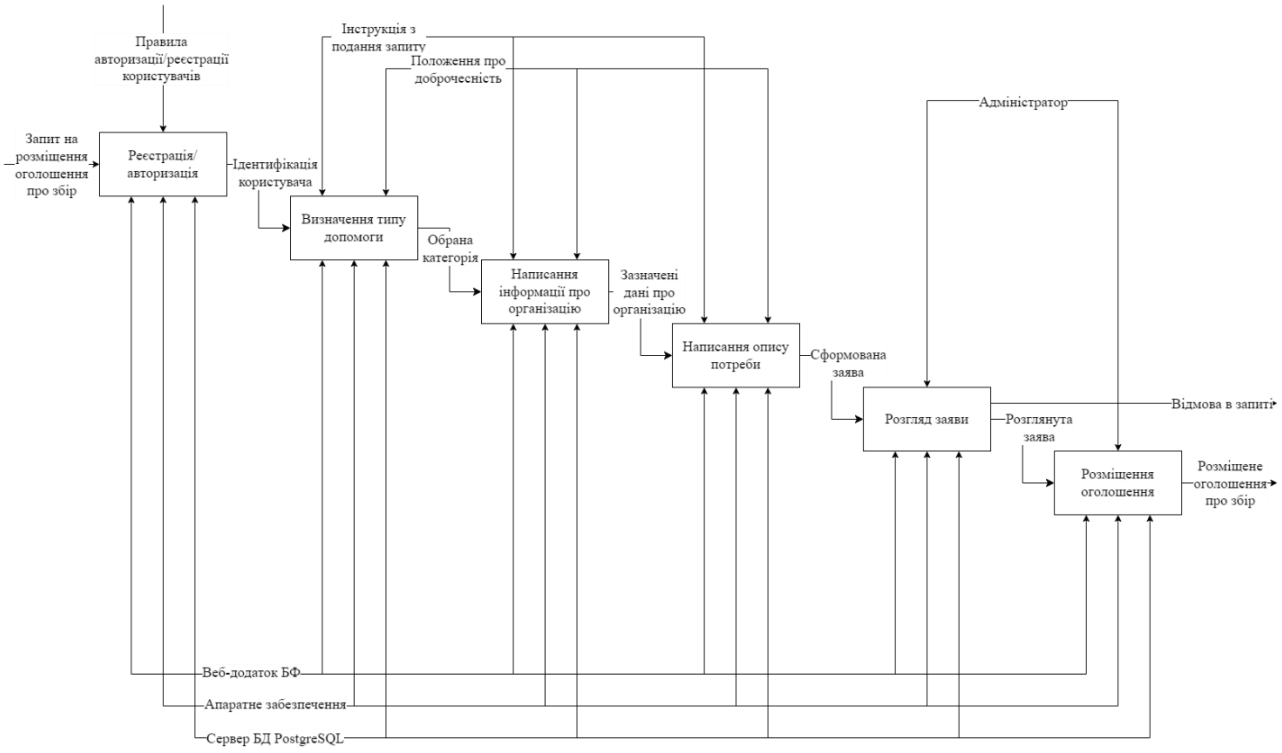

Рисунок 2.2 – Декомпозована діаграма процесу організації благодійної діяльності.

Таблиця 0.1 – Підпроцеси, входи, виходи, управління та механізми процесу організації благодійної діяльності.

Підпроцес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Реєстрація/ авторизація	Запит на розміщення оголошення про збір	Правила авторизації/реєстрації користувачів	Вебдодаток БФ, апаратне забезпечення, сервер БД PostgreSQL	Ідентифікація користувача
Визначення типу допомоги		Інструкція з подання запиту, положення про доброчесність		Обрана категорія
Написання інформації про організацію				Зазначені дані про організацію
Написання опису потреби				Сформована заява
Розгляд заяви		Адміністратор		Відмова в запиті
				Розглянута заява
Розміщення оголошення				Розміщене оголошення про збір

2.2 Модель послідовності виконання

Для демонстрації послідовності додавання оголошення про збір створено діаграму послідовності (Sequence Diagram) [11] (Рисунок 2.). У ній присутні два актори: користувач та адміністратор, а також веб-додаток, панель адміністратора і база даних. Користувач надає запит адміністратору на розміщення збору через веб-додаток, і він отримує в результаті або сповіщення про відмову, або сповіщення про публікацію збору, тим часом адміністратор у випадку задоволення запиту через адміністративну панель додає до бази даних інформацію про збір.

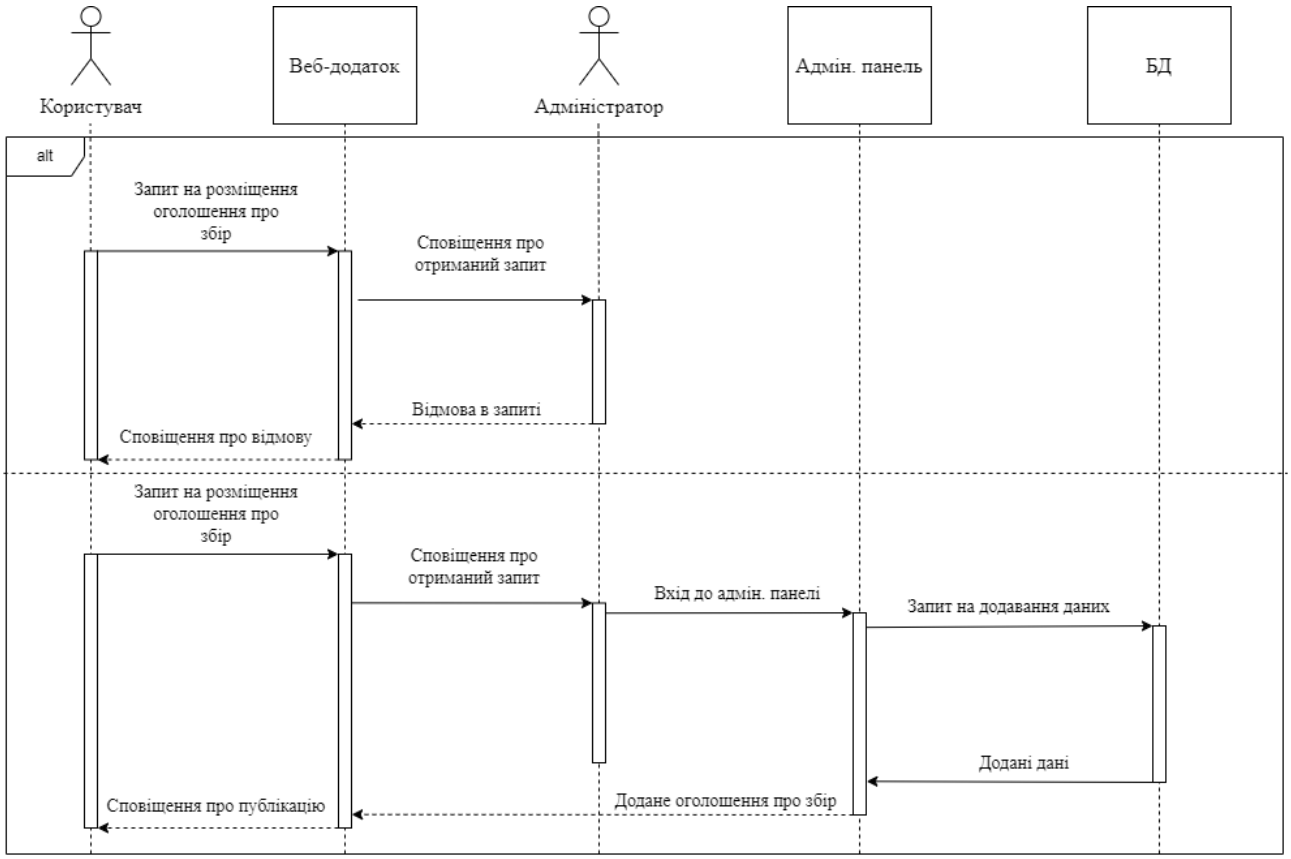

Рисунок 2.3 – Діаграма послідовності додавання збору.

2.3 Модель варіантів використання

Після створення моделі процесів та структури майбутнього вебзастосунку необхідно створити діаграму варіантів використання (Use Case Diagram) [12]. Вона визначає ролі та їхню взаємодію з системою.

Діаграма складається з двох основних компонентів: акторів та варіантів використання. Акторами в розроблюваному програмному продукті є гість, користувач, адміністратор та база даних (БД).

У майбутній системі розглядаються такі варіанти використання:

- реєстрація;
- авторизація;
- перегляд зборів;
- перегляд звітів;
- перегляд даних про автора збору;

- пожертвування грошей на благодійність;
- отримання сповіщень;
- створення запитів на розміщення зборів;
- розміщення оголошень про надання допомоги/потреби в допомозі (створення/редагування/видалення оголошень);
- розміщення зборів (створення/редагування/видалення зборів);
- підтвердження/відхилення запитів.

Діаграма варіантів використання майбутнього веб-додатку наведена на Рисунок 2..

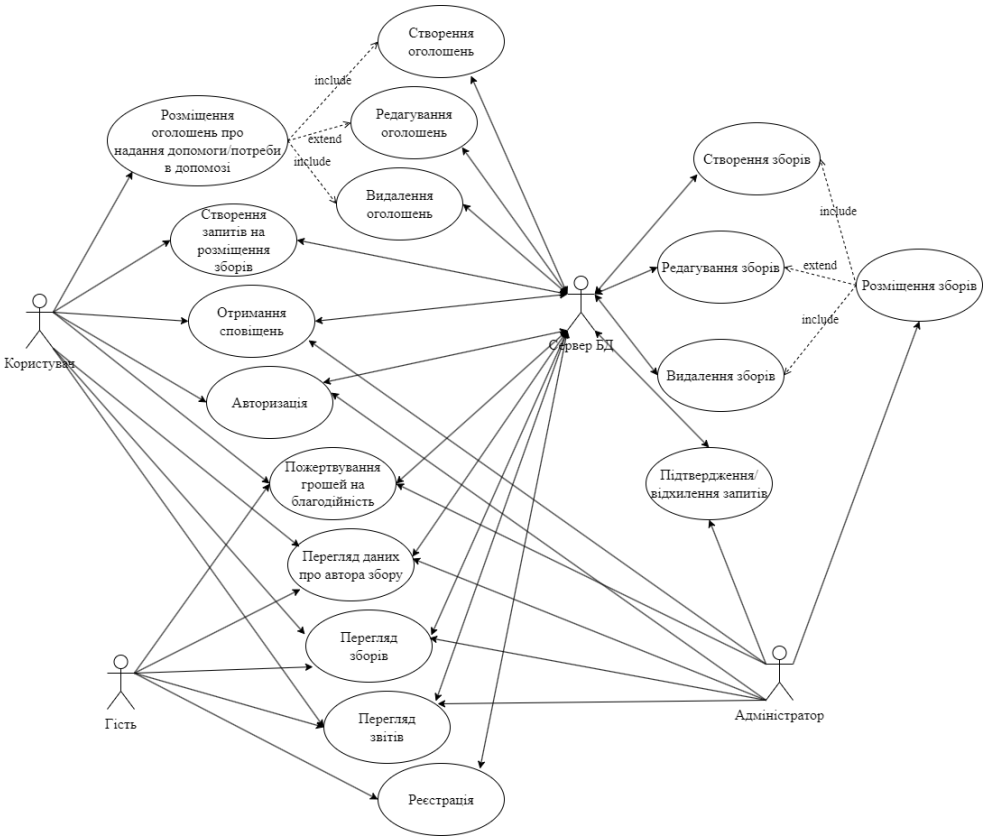

Рисунок 2.4 – Діаграма варіантів використання веб-застосунку благодійного фонду.

2.4 Структура бази даних

Для додавання, зміни, зберігання та видалення даних у вебзастосунку вирішено підключити базу даних, яка міститиме дані про користувачів, збори,

оголошення, звіти, медіа тощо.

База даних міститиме наступні 7 сутностей:

- user – дані користувачів;
- avatars – фото облікових записів користувачів;
- admin – дані адміністраторів та кандидатів;
- notice – дані оголошень про надання допомоги;
- fundraise – дані зборів;
- result – дані звітів по зборах;
- photo – фотографії оголошень, зборів та звітів.

Зв’язки між сутностями наведено на Рисунок 2., та перелік власне сутностей з описом полів наведено на таблиці 2.2:

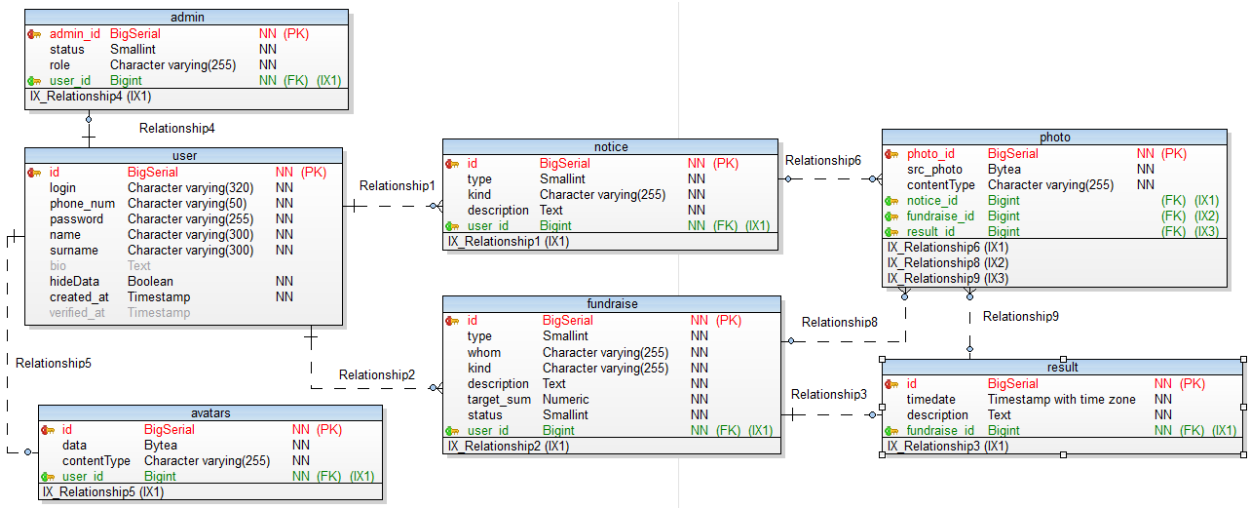

Рисунок 2.5 – Структура бази даних веб-застосунку.

Таблиця 2.2 – Перелік сутностей та опис полів

Сутність	Поле	Значення	Тип даних	Ключі/обмеження
user	id	Номер користувача	BigSerial	Primary key, autoincrement
	login	Електронна адреса	Varchar(320)	Not null, Unique
	phone_num	Номер телефону	Varchar(50)	Not null
	password	Пароль	Varchar(255)	Not null
	name	Ім’я	Varchar(300)	Not null
	surname	Прізвище	Varchar(300)	Not null

Продовження таблиці 2.2

Сутність	Поле	Значення	Тип даних	Ключі/обмеження
user	bio	Про себе	Text	-
	hideData	Приховати дані	Boolean	Not null
	createdAt	Дата та час створення	Timestamp	Not null
	verifiedAt	Дата та час підтвердження	Timestamp	-
avatars	id	Номер фото	BigSerial	Primary key, autoincrement
	data	Фото	Bytea	Not null
	contentType	Розширення	Varchar(255)	Not null
	user_id	Номер користувача	BigSerial	Foreign key, not null
admin	id	Номер адміністратора	BigSerial	Primary key, autoincrement
	status	Статус розглянутої заявки	Smallint	Not null
	role	Роль	Varchar(255)	Not null, default “Адміністратор”
	user_id	Номер користувача	BigSerial	Foreign key, not null
notice	id	Номер оголошення	BigSerial	Primary key, autoincrement
	type	Тип допомоги	Smallint	Not null
	kind	Вид допомоги	Varchar(255)	Not null
	description	Опис	Text	Not null
	user_id	Номер користувача, який додав	BigSerial	Foreign key, not null
fundraise	id	Номер збору	BigSerial	Primary key, autoincrement
	type	Тип збору	Smallint	Not null
	whom	Кому призначений збір	Varchar(255)	Not null
	kind	Вид збору	Varchar(255)	Not null
	description	Опис	Text	Not null
	target_sum	Ціль збору	Numeric	Not null
	status	Статус розглянутої заявки	Smallint	Not null
	user_id	Номер користувача, який надав запит	BigSerial	Foreign key, not null

Продовження таблиці 2.2

Сутність	Поле	Значення	Тип даних	Ключі/обмеження
result	id	Номер звіту	BigSerial	Primary key, autoincrement
	timedate	Час публікації	Timestamp	Not null
	description	Опис звіту	Text	Not null
	fundraise_id	Номер збору	BigSerial	Foreign key, not null
photo	photo_id	Номер фото	BigSerial	Primary key, autoincrement
	src_photo	Фото	Bytea	Not null
	contentType	Розширення	Varchar(255)	Not null
	notice_id	Номер оголошення	BigSerial	Foreign key
	fundraise_id	Номер збору	BigSerial	Foreign key
	result_id	Номер звіту	BigSerial	Foreign key

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Архітектура створюваного додатку

Створюваний вебзастосунок використовує клієнт-серверну архітектуру, принцип роботи якого зображений на рисунку 3.1 [13].

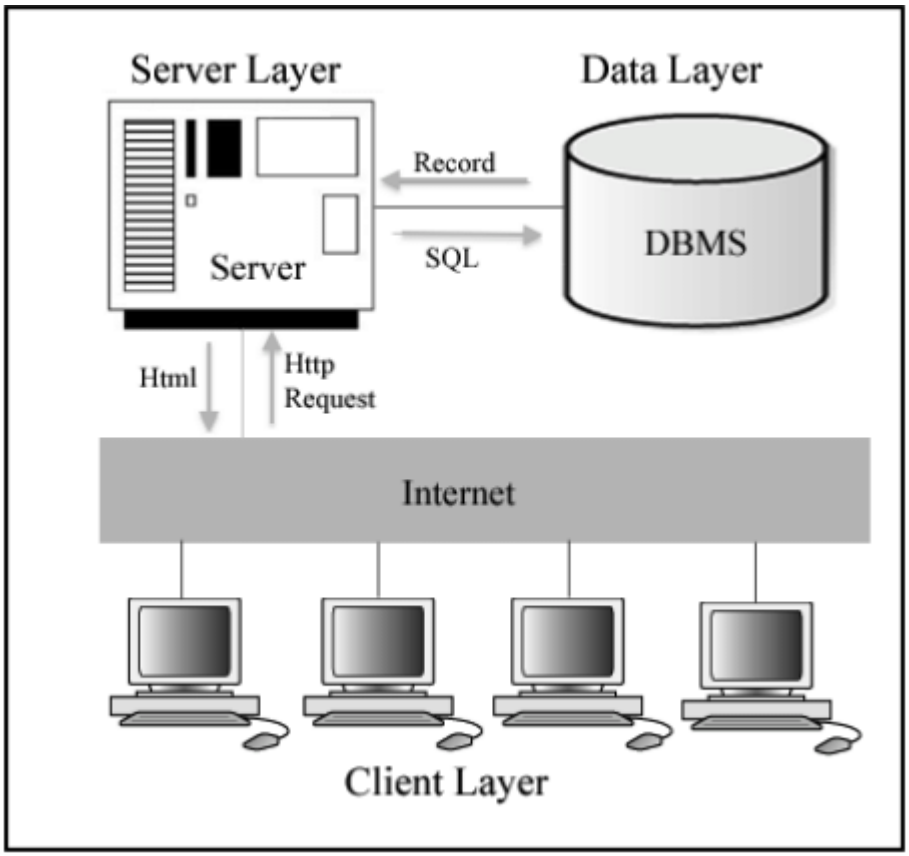

Рисунок 3.1 – Схема роботи клієнт-серверної архітектури.

Як можна побачити з наведеного рисунку, робота обраної архітектури відбувається за наступним принципом:

1) користувач за допомогою веб-браузера надсилає запит до веб-серверу;

2) веб-сервер оброблює отриману інформацію із запиту та надсилає відповідь, яка може описувати як успіх виконаної операції, так і про помилку, яка може

3) користувач отримує відповідь, яка відображається у веб-браузері.

Також у цій архітектурі може бути задіяним сервер бази даних, який

містить дані про користувачів чи інші дані. У такому разі веб-сервер надсилає запит до БД на додавання, отримання, редагування чи видалення даних, і сервер БД надсилає відповідь про те, чи знайдено необхідні записи, або про успішність чи неуспішність виконання операцій додавання, редагування чи видалення даних.

3.2 Вибір засобів програмної реалізації

Для розробки вебдодатку обрано такі основні технології:

- React.js [14] – фронтенд-бібліотека JavaScript для створення користувацьких інтерфейсів або його компонентів.

Дану технологію обрано через те, що має наступні переваги:

1. React робить можливим проєктувати динамічні вебзастосунки, так як не потребує багато кодування і надає розробнику більше функціональності, на відміну від звичайного JavaScript.
2. React потребує для роботи віртуальну об’єктну модель документа, тим самим створюючи веб-додатки швидше.
3. Компоненти в React є фундаментальною складовою розробки, адже їх можна перевикористовувати у самому додатку, і це в свою чергу зменшує період розробки додатків.
4. Для розробки необхідно вкладати дочірні модулі всередину батьківських компонентів. Завдяки тому, що потік даних у React відбувається в одному напрямку, це надає можливість розробнику швидше знаходити місце, де виникає помилка, а отже робить відладку додатку легшою.
5. Може використовуватися як для веб-розробки, так і для створення мобільних додатків.

- Bootstrap [15] – CSS-фреймворк для розробки респонсивних та мобільно-орієнтовних вебсайтів.
- Node.js [16] – JavaScript кросплатформенне середовище виконання. Він дозволяє створювати як клієнтську, так і серверну частину вебзастосунку.

- Express.js [16] – мінімальний та гнучкий Node.js фреймворк, призначений для розробки серверної частини веб-додатку. Однією з основних переваг даного інструменту полягає в тому, що він написаний мовою JavaScript, а отже дозволяє розробникам, які знайомі з цією мовою, не вивчати додатково інші мови програмування, тим самим роблячи бекенд розробку набагато легшим для фронтенд-розробників. Саме тому її і обрано для створення вебзастосунку благодійного фонду.
- PostgreSQL [17] – реляційна система керування базами даних.

Серед переваг даної СКБД можна виділити наступні:

1. Є open source системою, тобто не вимагає попередньої реєстрації чи оплати.
2. Сумісна з багатьма мовами програмування, включно з JavaScript.
3. Підтримує останні стандарти мови SQL, маючи 170 із 179 обов'язкових особливостей.
4. Є високо розширюваною, тобто дозволяє визначати власні типи даних, писати власні функції, писати код з різних мов програмування без необхідності перезапуску бази даних.

Окрім перелічених технологій у роботі використано наступні додаткові бібліотеки:

1. Клієнтська частина:

1.1.react-router-dom [18] – бібліотека для створення навігації у вебзастосунку, яка містить такі компоненти, як BrowserRouter, Routes, Route, NavLink, Navigate, а також хук useNavigate, який дозволяє переміщатися з однієї сторінки на іншу.

1.2.react-bootstrap [19] – бібліотека, яка спрощує використання Bootstrap компонентів, конденсуючи оригінальний Bootstrap у компоненти у стилі React.

1.3.react-avatar [20] – призначене для портретів профілів користувача, основним компонентом якого є Avatar.

1.4.axios [21] – клієнт HTTP на основі Promise для Node.js та браузера. Надає можливість перехопити запити та відповіді, перетворювати дані запиту

та відповіді, скасування запитів.

2. Серверна частина

2.1.cors [22] – механізм, що підтримує крос-доменні запити та передачу даних між браузером та веб-сервером через захищене з’єднання.

2.2.bcrypt [23] – бібліотека для хешування паролів.

2.3.sequelize [24] – бібліотека, призначена для роботи з базами даних. Надає можливість користувачеві при запуску сервера додатку одразу створити таблиці та зв’язки між ними. Також дана бібліотека дозволяє виконувати операції додавання, редагування, пошуку та видалення даних без написання запитів мовою SQL.

2.4.dotenv [25] – бібліотека, призначена зчитувати дані змінних середовища, що містяться у файлах типу .env, у process.env.

2.5.express-validator [26] – бібліотека для перевірки на валідність введених даних, наприклад email.

2.6.jsonwebtoken [27] – призначена для створення токенів доступу, які використовуються для передачі даних для автентифікації, і підписуються секретним ключем.

2.7.cookie-parser [28] – необхідне для зчитування даних, що знаходяться в cookie.

2.8.multer [29] – необхідне для роботи з файлами, які передаються зі сторони клієнта в сторону веб-сервера для подальшої обробки та додавання до бази даних.

2.9.nodemailer [30] – бібліотека, що призначена для надсилання листів користувачам.

2.10. nodemon – необхідне для запуску веб-серверу. Для цього після його завантаження у файлі package.json всередині блоку scripts варто прописати наступний рядок:

"dev": "nodemon index.js"

Також для проєктування та розробки вебдодатку використано наступні інструменти:

1. Visual Studio Code [31] – інтегроване середовище розробки, використане для написання клієнтської та серверної частини додатку.
2. GitHub – використаний для розміщення проєкту.
3. Toad Data Modeller – додаток для створення діаграм відношення сутностей.
4. Draw.io – веб-додаток для створення функціональних діаграм, діаграм послідовності та варіантів використання.
5. ChatGPT – для генерування текстів.

3.3 Опис програмної реалізації

3.3.1 Підготовка до роботи

Так як проєкт складатиметься з двох частин: клієнтської та серверної, тоді необхідно створити папку проєкту:

mkdir bachelor_work_final

cd bachelor_work_final

Наступним кроком створимо React проєкт шляхом виконання наступних команд:

npm init react-app react_bachelor_work

cd react_bachelor_work

Тобто після виконання наступної команди одразу створиться React проєкт з необхідними для роботи папками та файлами, такі як:

- папка public – містить логотип, метадані, маніфест та файл index.html;
- папка src, що містить компоненти, сторінки, файл з константами, фото, що використовуються в додатку, і головний компонент додатку App.js.

Після цього необхідно завантажити необхідні для подальшої роботи бібліотеки, про які згадано в пункті 3.2 Вибір засобів програмної реалізації:

npm i react-router-dom bootstrap react-bootstrap react-avatar axios

Також варто глобально встановити бібліотеку dotenv, задля зчитування значень змінних середовища, що містяться у файлі .env:

npm install -g dotenv

Додатково всередині папки src необхідно створити ще 4 папки за допомогою команди mkdir:

- pages: містить компоненти сторінок. Кожна сторінка містить окрему папку, в якій є файл компоненту з розширенням .js, та файл з розширенням .css для стилю сторінки, який потім імпортується до файлу сторінки. Якщо сторінка має інші компоненти, то вони містяться в папці components разом з підкомпонентами, які потім також імпортуються;
- components: містить компоненти, що використовуються в самих сторінках (хедер, футер, форми тощо). Кожний компонент, як і компонент сторінки, містить окрему папку, де містяться файл компоненту .js та файл стилів .css, а також папки з підкомпонентами;
- resources: містить фото, що використовуються в додатку;
- utils: містить константи, що використовуються в проєкті.

cd src

mkdir “pages”, “components”, “resources”, “utils”

Усередині цієї ж папки потрібно додати файл routes.js, який міститиме шляхи до сторінок та компоненти сторінок.

echo >routes.js

У папці React проєкту додатково варто додати файл .env, яка міститиме змінну, що посилатиметься на адресу веб-серверу. Таким чином, React проєкт міститиме структуру, зображену на рисунку 3.2.

cd ..

echo >.env

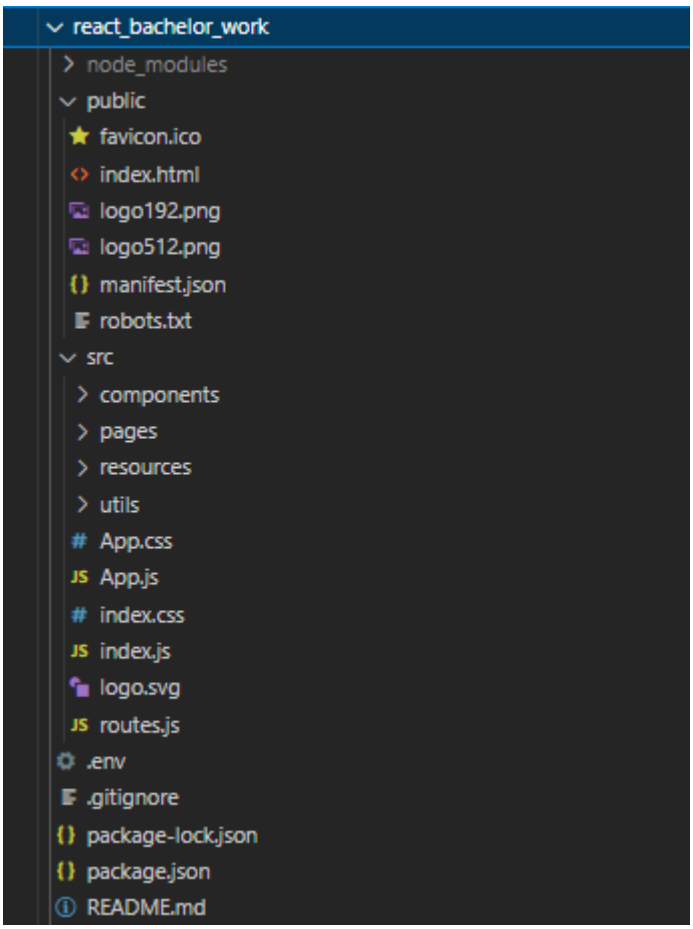

Рисунок 3.2 – Структура React проєкту.

Наступним етапом створимо та ініціалізуємо Express.js проєкт. Для цього створимо окрему папку express_bachelor_work та ініціалізуємо за допомогою наступних команд.

cd ..

mkdir express_bachelor_work

cd express_bachelor_work

npm init -y

echo >index.js

Для роботи необхідно встановити за допомогою npm і бібліотеки, про які згадано в пункті 3.2 Вибір засобів програмної реалізації:

npm i express cors sequelize express-validator jsonwebtoken cookie-parser multer nodemailer

Усередині Express.js проєкту також необхідно додати наступні файли та

папки:

- .env – файл зі змінними середовища, які містять номер порту, до якого підключатиметься веб-сервер, дані для підключення до бази даних, адреса електронної пошти, з якого надсилатимуться повідомлення, а також адресу, з якого отримуватимуться запити від клієнтської частини;
- db.js – файл підключення до БД;
- папка models: містить файл models.js, в якому описано класи сутностей, поля та зв’язки між сутностями;
- папка routes, в якому описуватимуться шляхи та послідовності обробки даних, які надходять з боку клієнта;
- папка controllers, що містить класи контролерів, які оброблюватимуть дані із запитів та відправлятимуть відповідь;
- папка error, що містить класи помилок;
- папка middleware, що містить файли, які дозволяють перед обробкою сервером даних перевірити запит, і якщо дані є невалідними, то повернути повідомлення про помилку.

mkdir “models”, “routes”, “controllers”, “error”, “middleware”

echo >.env

echo >db.js

Отже, структура Express.js проєкту виглядатиме, як на рисунку 3.3.

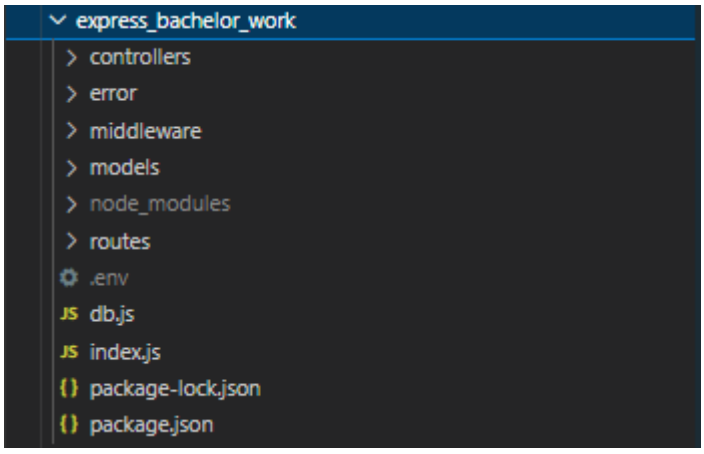

Рисунок 3.3 – Структура Express.js проєкту.

Для запуску React проєкту використовується команда

npm start

Для запуску Express.js проєкту:

npm run dev

3.3.2 Використання компонентів React

Як і сказано в пункті 3.2 Вибір засобів програмної реалізації, React обрано через те, що компоненти можна перевикористовувати у залежності від того, яка інформація передається в нього, та який стан має власне програма. Один із способів це зробити – передавати до компоненту props. Вони є параметрами, які можуть змінювати поведінку та вигляд самого компоненту, що робитиме його перевикористаним декілька разів.

Прикладом цього може слугувати сторінка з часто задаваними питаннями FAQ.js, яка приймає на вхід масив об’єктів, кожен з яких містить питання та відповідь:

```
function FAQ(props) {
   return(
      <main>
         <Container className="mt-3">
            <h2>Часто задавані питання</h2>
         </Container>
         <Accordion>
            {props.quests.map((quest, index) =>
               <Accordion.Item key={index} eventKey={'' + index}>
                  <Accordion.Header>{quest.question}</Accordion.Header>
                  <Accordion.Body>
                     {quest.answer}
                  </Accordion.Body>
               </Accordion.Item>
            )}
         </Accordion>
      </main>
   );
}
```

У цьому випадку компонент сторінки приймає усе, що передається у властивості, включно з масивом об’єктів. Тобто у залежності від того, скільки

об’єктів матиме масив, стільки і відображатиметься питань та відповіді на них.

Ще один із варіантів перевикористання одних і тих же компонентів є умовний рендеринг. Тобто завдяки умовам, таким як: чи є користувач авторизованим, або чи знайдено дані, знайдені з використанням фільтрів тощо, сторінка відображається по-різному.

Приклад використання продемонстровано в компоненті NavigationPanel.js, який у залежності від того, чи є користувач авторизованим чи ні, і якщо так, то чи не є він також адміністратором, відображає різні кнопки.

```
{
   isLoggedIn ?
   <Nav className="nav navbar-right">
      <NavLink to={USER_ROUTE} className="nav-link lat">
         <Avatar alt="Профіль" src={avatar !== props.avatar ? `data:${avatar.contentType};base64,${avatar.data}` : props.avatar} size="2.4em" />
      </NavLink>
      {
         isAdmin ?
         <NavLink to={ADMIN_ROUTE} className="nav-link lat">
            <Button type="button" className="btn btn-primary my-2 rob-btn">Адмін панель</Button>
         </NavLink>
         :
         <></>
      }
      <NavLink to={LANDING_ROUTE} className="nav-link lat">
         <Button type="button" className="btn btn-warning my-2 rob-btn" onClick={logOut}>Вийти</Button>
      </NavLink>
   </Nav>
   :
   <Nav className="nav navbar-right">
      <NavLink to={LOGIN_ROUTE} className="nav-link lat">
         <Button type="button" className="btn btn-outline-light me-2 my-2 rob-btn">Увійти</Button>
      </NavLink>
      <NavLink to={SIGNUP_ROUTE} className="nav-link lat">
         <Button type="button" className="btn btn-warning my-2 rob-btn">Зареєструватися</Button>
```

```
        </NavLink>
    </Nav>
}
```

Тобто якщо користувач неавторизований, він бачитиме кнопки реєстрації та авторизації, інакше бачитиме кнопку входу до профілю та виходу з облікового запису, і якщо є ще адміністратором, то бачитиме кнопку входу до адмінпанелі.

3.3.3 Маршрутизація

React має бібліотеку React Route, що допомагає застосувати до динамічного вебзастосунку маршрутизацію, яка дозволяє користувачам безперебійно переміщатися по сторінках [14].

Основними компонентами, що застосовані в проєкті, є BrowserRouter, який застосовано всередині головного компоненту App.js і містить у собі набори шляхів, які описуються за допомогою компоненту Routes. Усередині Routes містяться компоненти Route, які визначають шляхи від головної сторінки до інших сторінок, і якщо користувач вирішив переміститися з однієї сторінки на іншу, то відображатиметься конкретний компонент сторінки. Якщо користувач увів неправильний шлях, то для цього використовується компонент Navigate, який дозволить одразу перемістити користувача на головну.

Повний опис того, як застосовується маршрутизація в додатку, наведено у файлах App.js, AppRouter.js та NavigationPanel.js у Додатку А.

Структура шляхів у проєкті React наведено на рисунку 3.4.

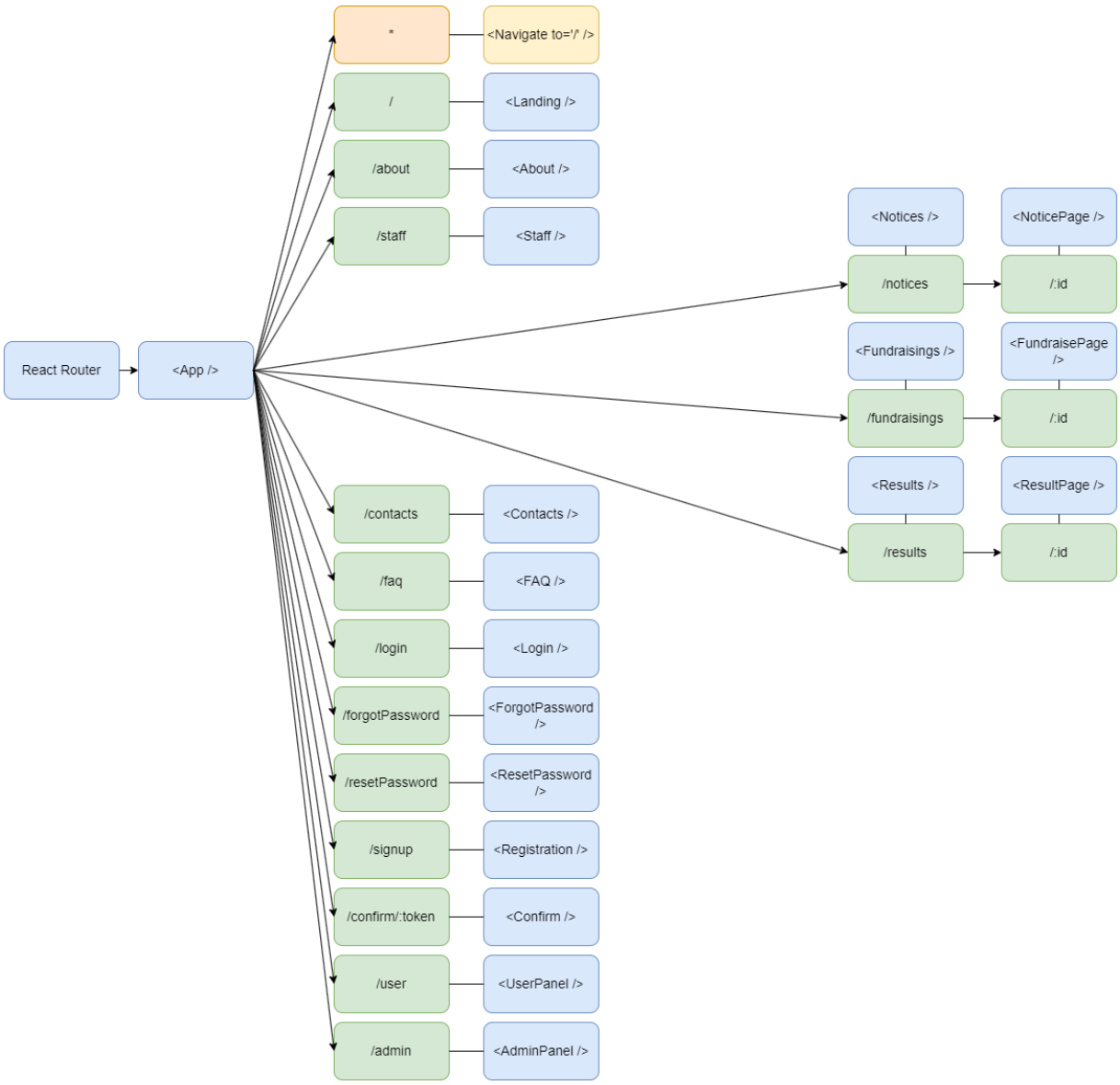

Рисунок 3.4 – Структура шляхів у вебзастосунку.

3.3.4 Заходи забезпечення функціонування системи

Для того, щоб вебзастосунок безперебійно функціонував, а дані користувачів залишалися захищеними від несанкціонованого доступу, вжито наступні заходи:

1. У формах реєстрації та авторизації, якщо користувач не ввів необхідні дані, то кнопка вимикається, тобто стає недоступною для натиснення. Для прикладу наведено нижче кнопку, яка використовується на сторінці авторизації.

<Button variant="primary" type="submit" disabled={login === "" || password === ""} onClick={logIn}>

```
    Увійти
</Button>
```

2. Валідація введеного email відбувається на стороні серверу за допомогою бібліотеки express-validator, яка перед надсиланням запиту до контролеру перевіряє, чи введено електронну адресу, і потім передає результат перевірки до контролеру. Якщо знайдено помилку, то повертається помилка.

```
userRouter.js
/*робота userRouter*/
const { body } = require('express-validator');
/*робота userRouter*/
router.post('/login', body('login').isEmail(), userController.login);
/*робота userRouter*/
UserController.login
async login(req, res, next) {
    const { login, password } = req.body;

    try {
        const errors = validationResult(req);

        if (!errors.isEmpty()) {
            return next(ApiError.badRequest("Невалідний email!"));
        }
/*робота методу login*/
    } catch (e) {
        next(ApiError.badRequest(e.message))
    }
}
```

3. Хешування паролю відбувається перед додаванням даних користувача до бази даних. Для цього використано функцію bcrypt.hash, куди передаються такі параметри, як введений пароль та кількість хешувань, через які проходить пароль. При авторизації пароль порівнюється за допомогою bcrypt.compareSync, куди передається введений та зашифрований пароль. Якщо пароль введено неправильно, повертається повідомлення про помилку.

4. Для запобігання несанкціонованого доступу до профілю застосовано хук useEffect, який перевіряє, чи є користувач авторизованим. Якщо користувач не авторизований, то відбувається перенаправлення на головну сторінку.

```
axios.defaults.withCredentials = true;
useEffect(() => {
  axios.get(process.env.REACT_APP_API_URL + 'user/profile')
  .then(res => {
    if (res.data.status) {
      setUserData(res.data.user);

      if (res.data.user.verifiedAt === null) {
        setVerifiedData(false);
      }

      if (res.data.portrait !== null) {
        setAvatar(res.data.portrait);
      }
    } else {
      navigate(LANDING_ROUTE);
    }
  })
});
```

5. Для обробки помилок на стороні серверу використовується блок try-catch, завдяки якому при виявленні виняткової ситуації повертається повідомлення про помилку. Приклад наведено для функції авторизації користувача.

```
async login(req, res, next) {
    const { login, password } = req.body;

    try {
        const errors = validationResult(req);

        if (!errors.isEmpty()) {
            return next(ApiError.badRequest("Невалідний email!"));
        }

        const user = await User.findOne({where: {login}});
        if (!user) {
            return next(ApiError.badRequest("Користувач не зареєстрований!"));
        }

        let checkPassword = bcrypt.compareSync(password, user.password);
        if (!checkPassword) {
            return next(ApiError.badRequest("Пароль неправильний!"));
        }
```

```
            const accessToken = generateAccessJwt(user.login, user.phone_num, user.name, user.surname);
            const refreshToken = generateRefreshJwt(user.login, user.phone_num, user.name, user.surname);

            res.cookie('accessToken', accessToken, {maxAge: 900000});
            res.cookie('refreshToken', refreshToken, {maxAge: 3600000 * 8, httpOnly: true, secure: true, sameSite: 'strict'});

            return res.json({status: true, message: "Авторизація успішна!"});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }
```

3.4 Аналіз результатів

3.4.1 Початок роботи із сайтом

На початку роботи користувач бачить головну сторінку, наведену на рисунку 3.5. Для мобільних пристроїв головна сторінка матиме наступний вигляд, як на рисунку 3.6.

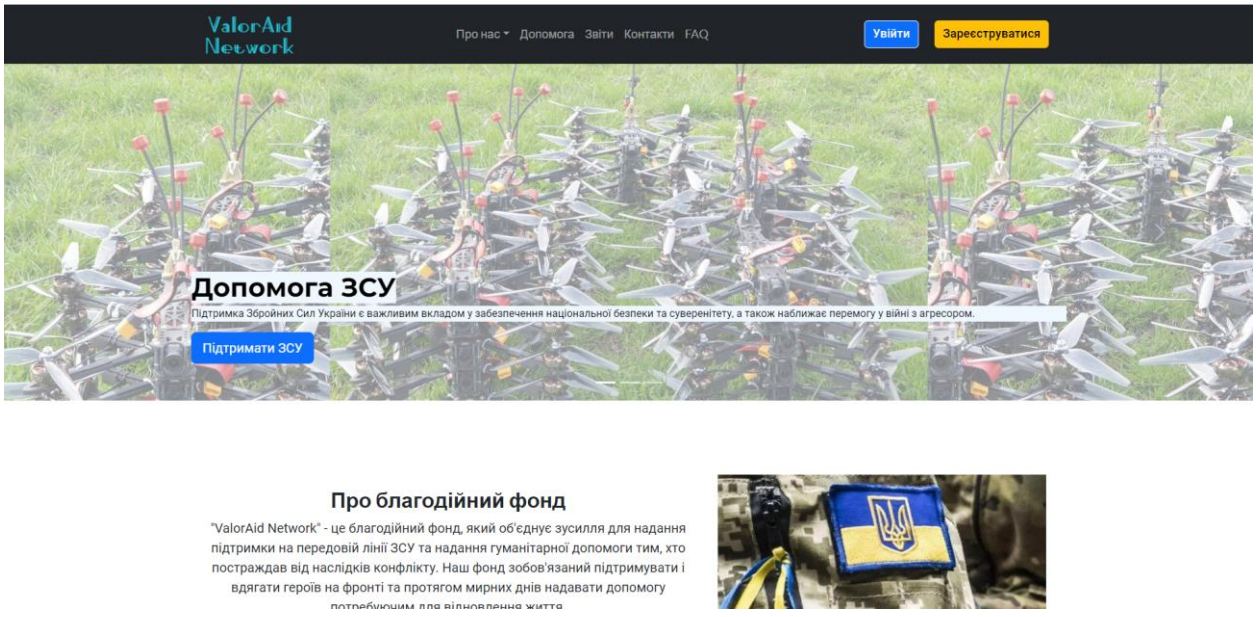

Рисунок 3.5 – Головна сторінка.

Рисунок 3.6 – Головна сторінка на мобільних пристроях.

Також на рисунках 3.7-3.15 можна побачити сторінки «Про фонд», контактів та часто задаваних питань, а також таких компонентів, як хедер і футер.

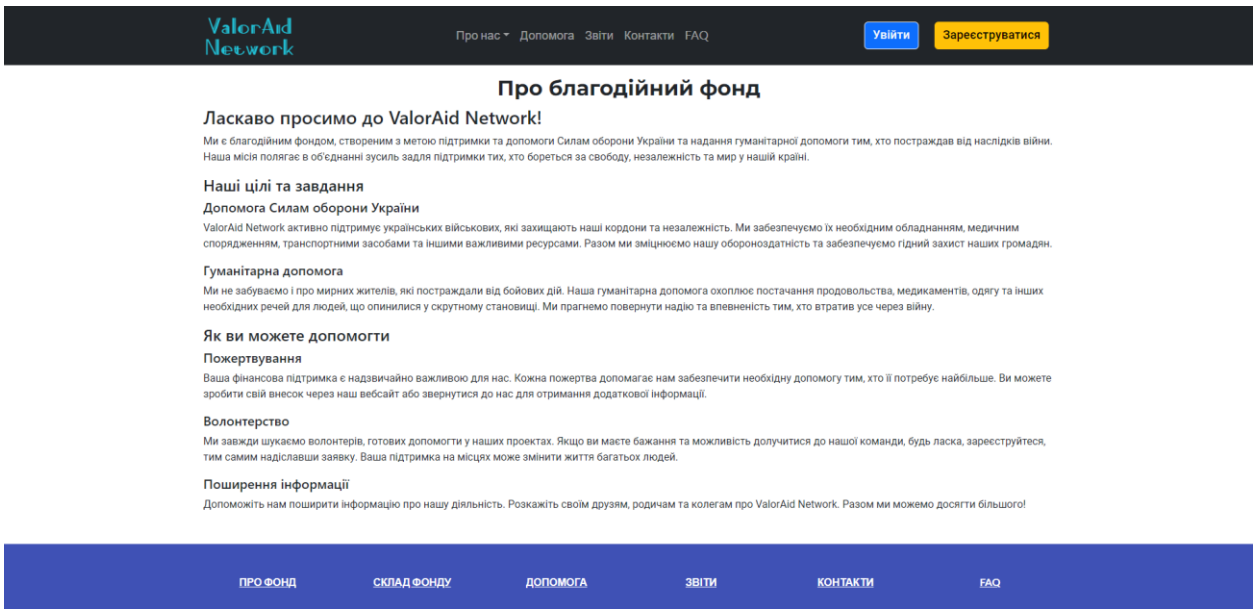

Рисунок 3.7 – Сторінка «Про фонд».

Рисунок 3.8 – Сторінка «Про фонд» на мобільних пристроях.

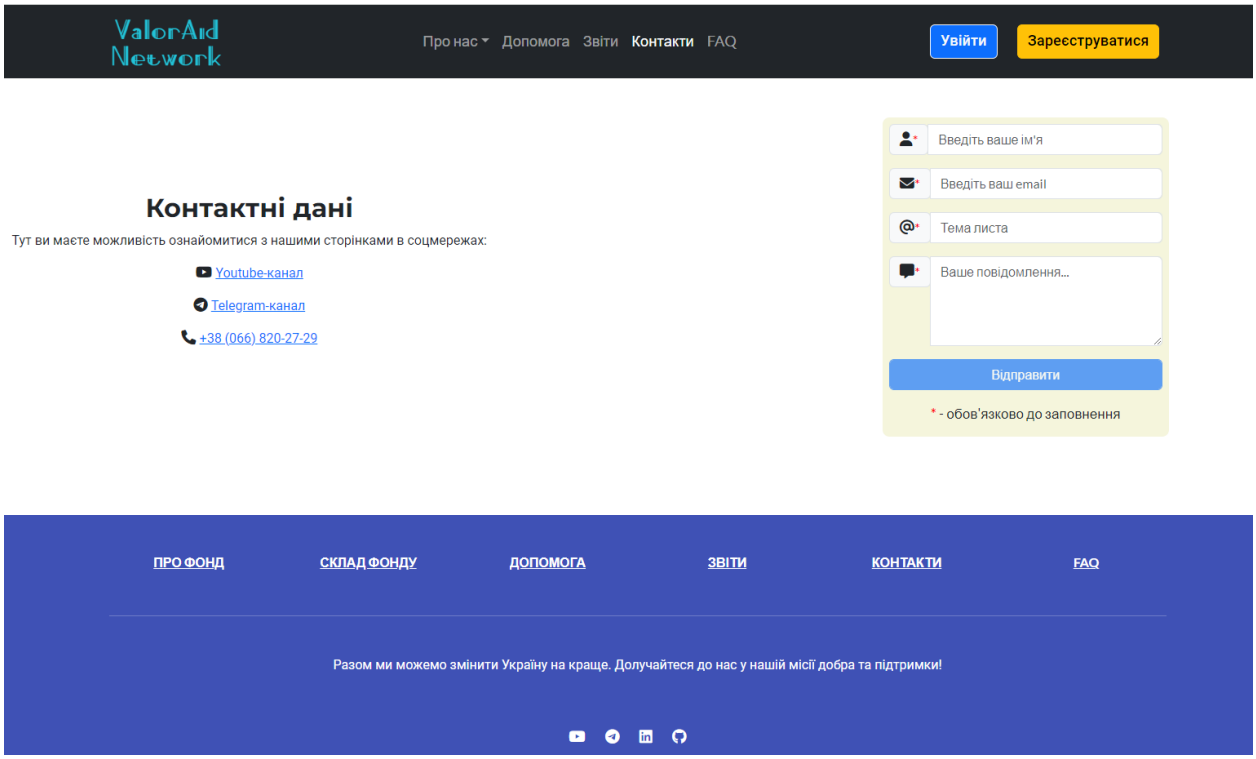

Рисунок 3.9 – Сторінка контактних даних.

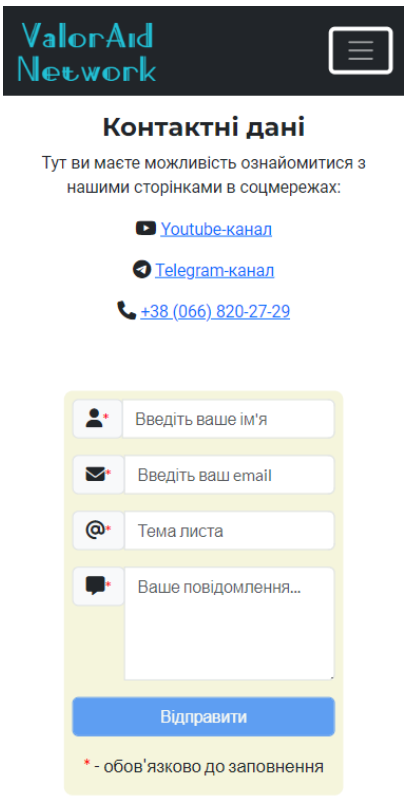

Рисунок 3.10 – Сторінка контактних даних.

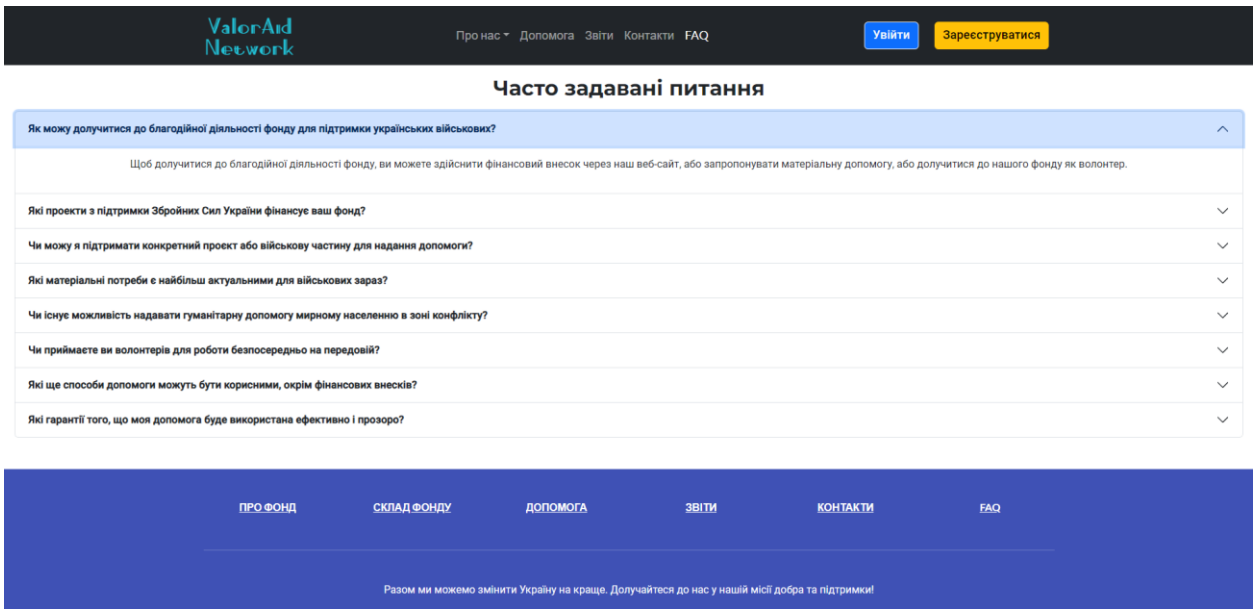

Рисунок 3.11 – Сторінка часто задаваних питань.

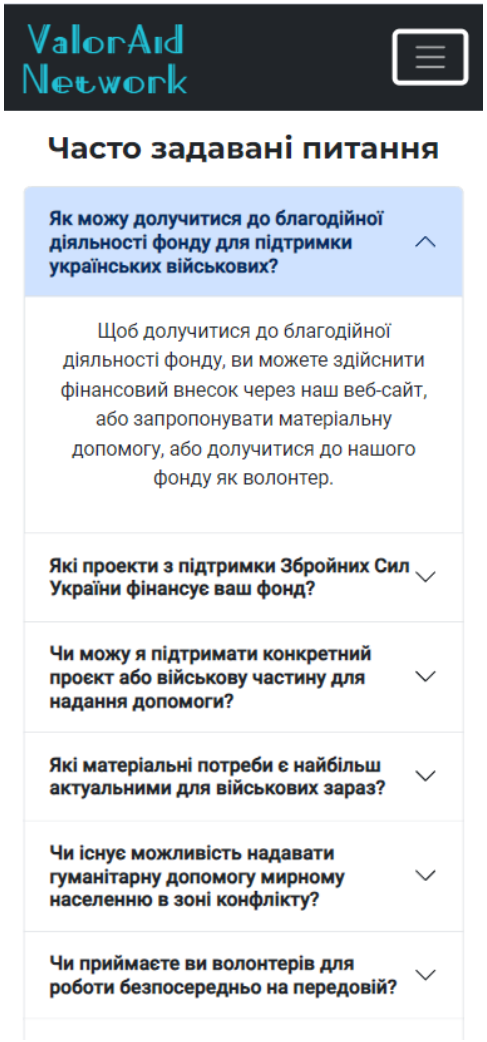

Рисунок 3.12 – Сторінка часто задаваних питань на мобільних пристроях.

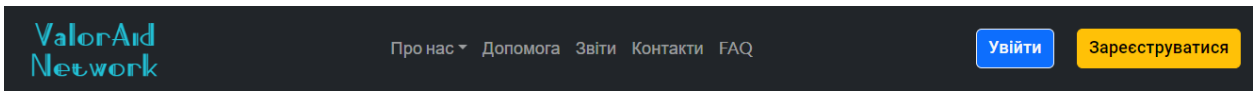

Рисунок 3.13 – Хедер.

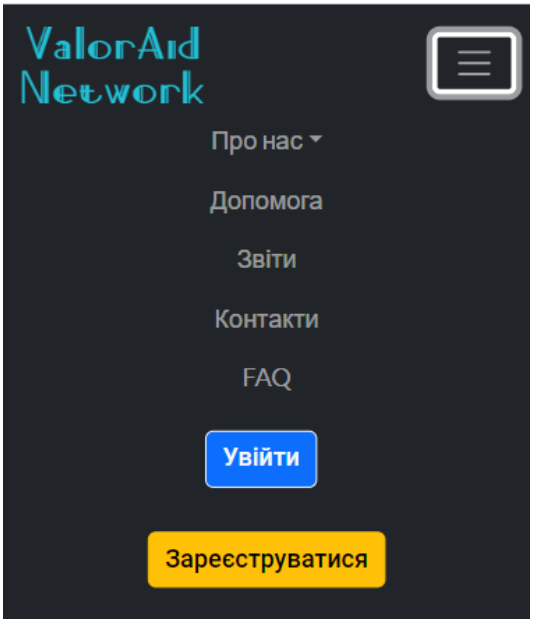

Рисунок 3.13 – Хедер на мобільних пристроях у розгорнутому вигляді.

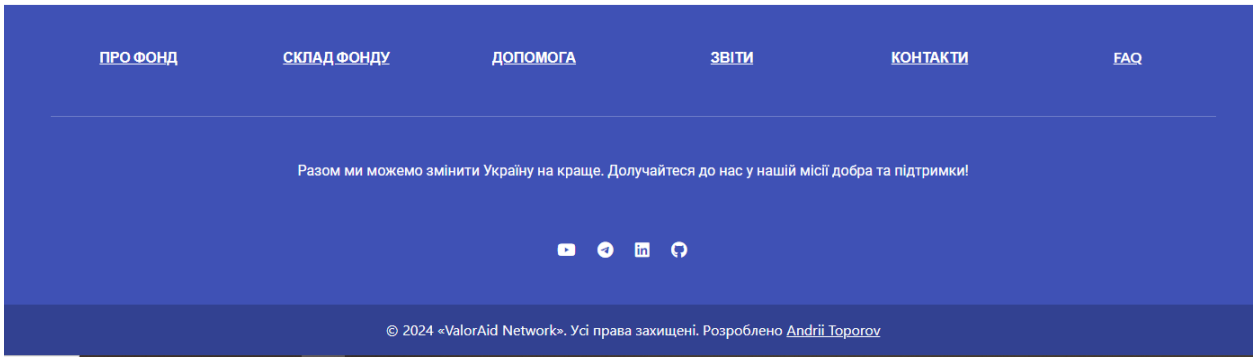

Рисунок 3.14 – Футер.

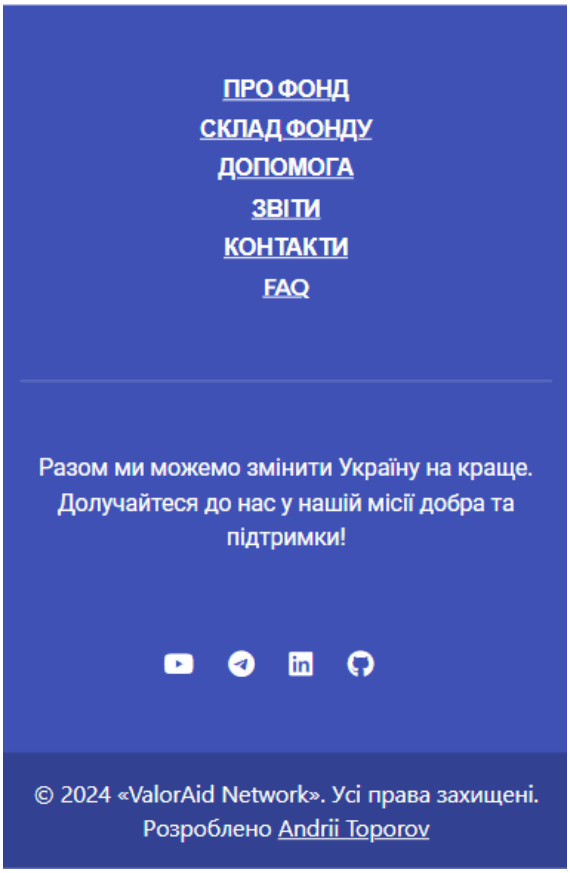

Рисунок 3.15 – Футер на мобільних пристроях.

3.4.2 Робота форми зворотнього зв’язку

Форма зворотнього зв’язку міститься як на головній сторінці (рисунок 3.16), так і на сторінці з контактними даними (рисунок 3.17). Вона доступна як для авторизованих користувачів, так і для неавторизованих.

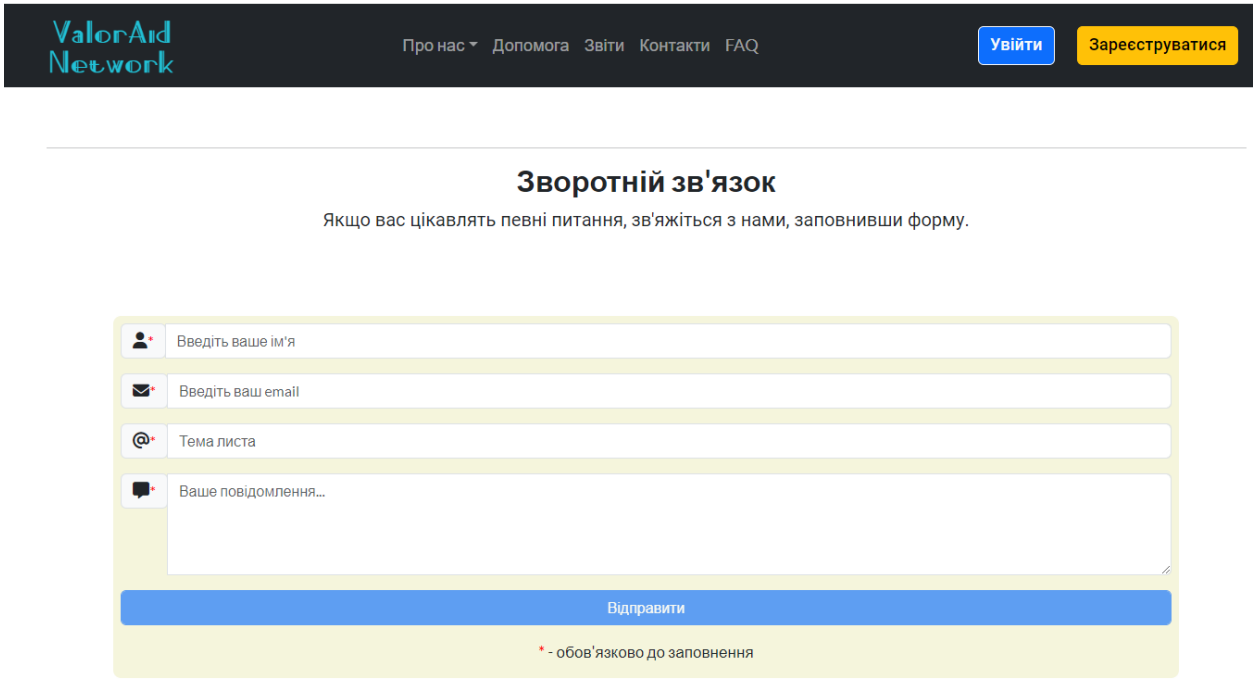

Рисунок 3.16 – Форма на головній.

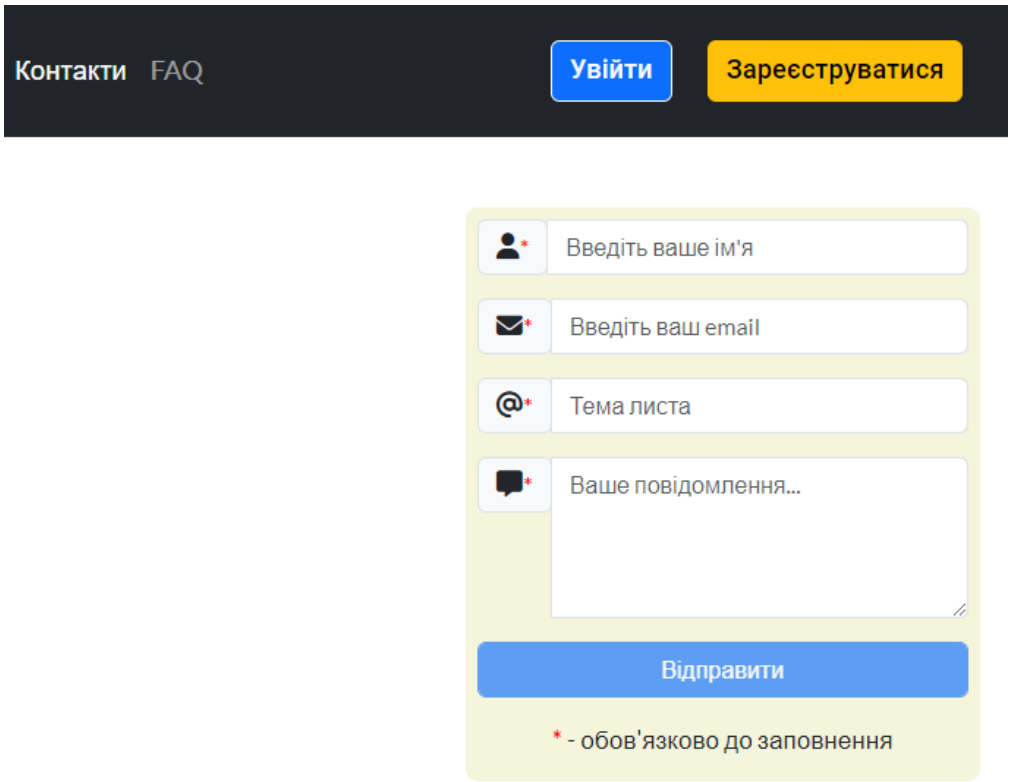

Рисунок 3.17 – Форма на сторінці контактних даних.

Для відправки повідомлення необхідно ввести ім’я, email, тему повідомлення та власне текст повідомлення. Після відправки з’являється сповіщення про відправку повідомлення (рисунок 3.18), а повідомлення надходить власнику (рисунок 3.19).

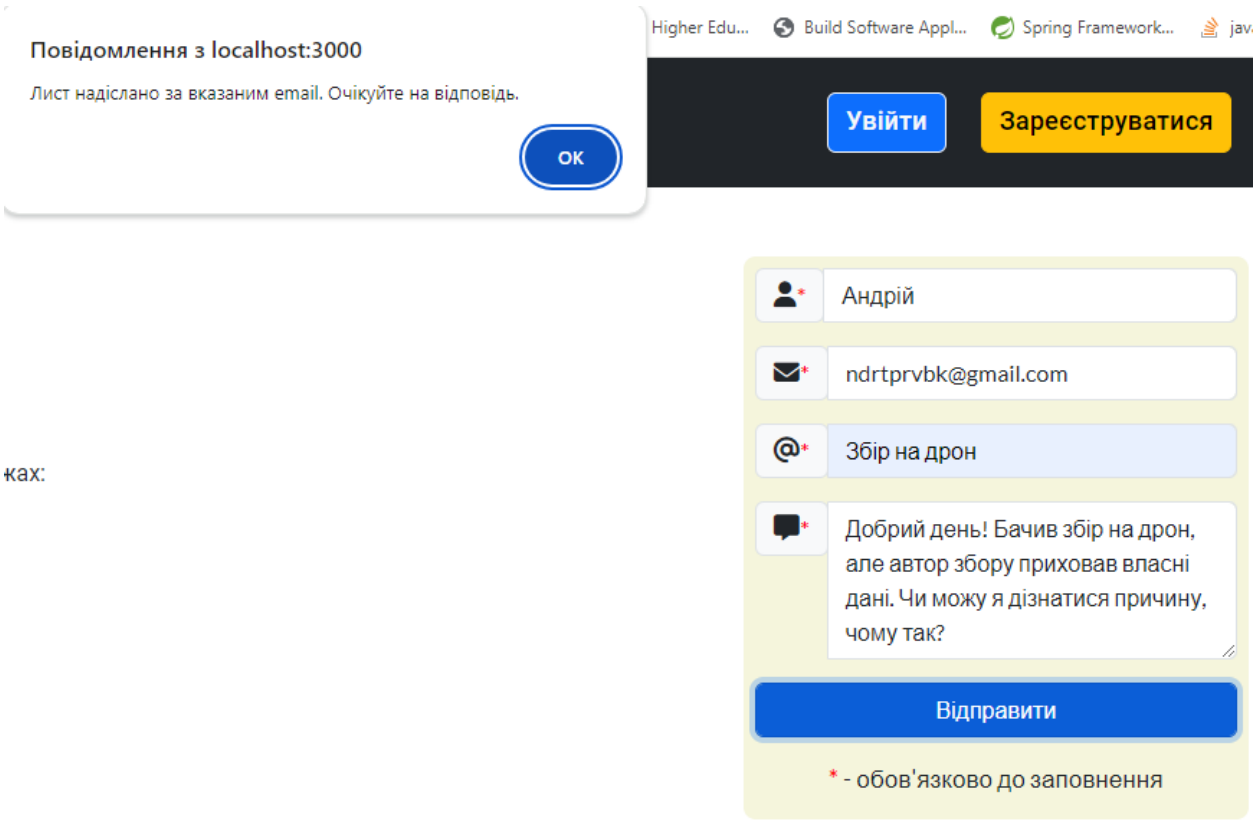

Рисунок 3.18 – Сповіщення про успішне відправлення.

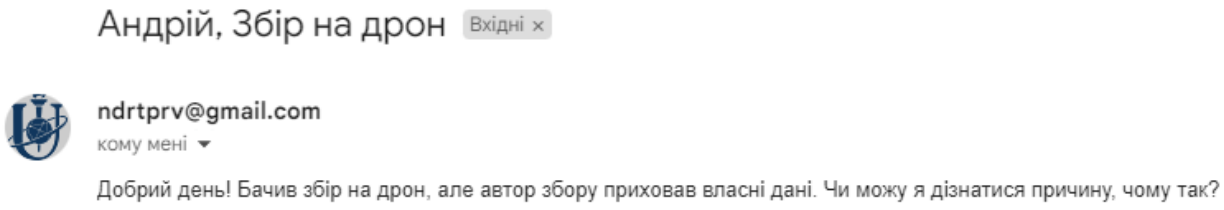

Рисунок 3.19 – Отримане повідомлення.

3.4.3 Реєстрація

Основним компонентом сторінки реєстрації є форма, яка складається з необхідних для заповнення полів, вибору ролі та визначення, чи варто приховати дані для інших користувачів. Як можна побачити з рисунку 3.20,

якщо користувач не ввів необхідні дані, не обрав роль, не визначився, чи приховувати дані від інших користувачів, чи ні, пароль не задовольняє вказаним умовам або значення паролю не збігається зі значенням у полі підтвердження паролю, то кнопка залишається заблокованою. Також ця форма є адаптованою під мобільні гаджети, що можна побачити на рисунку 3.21.

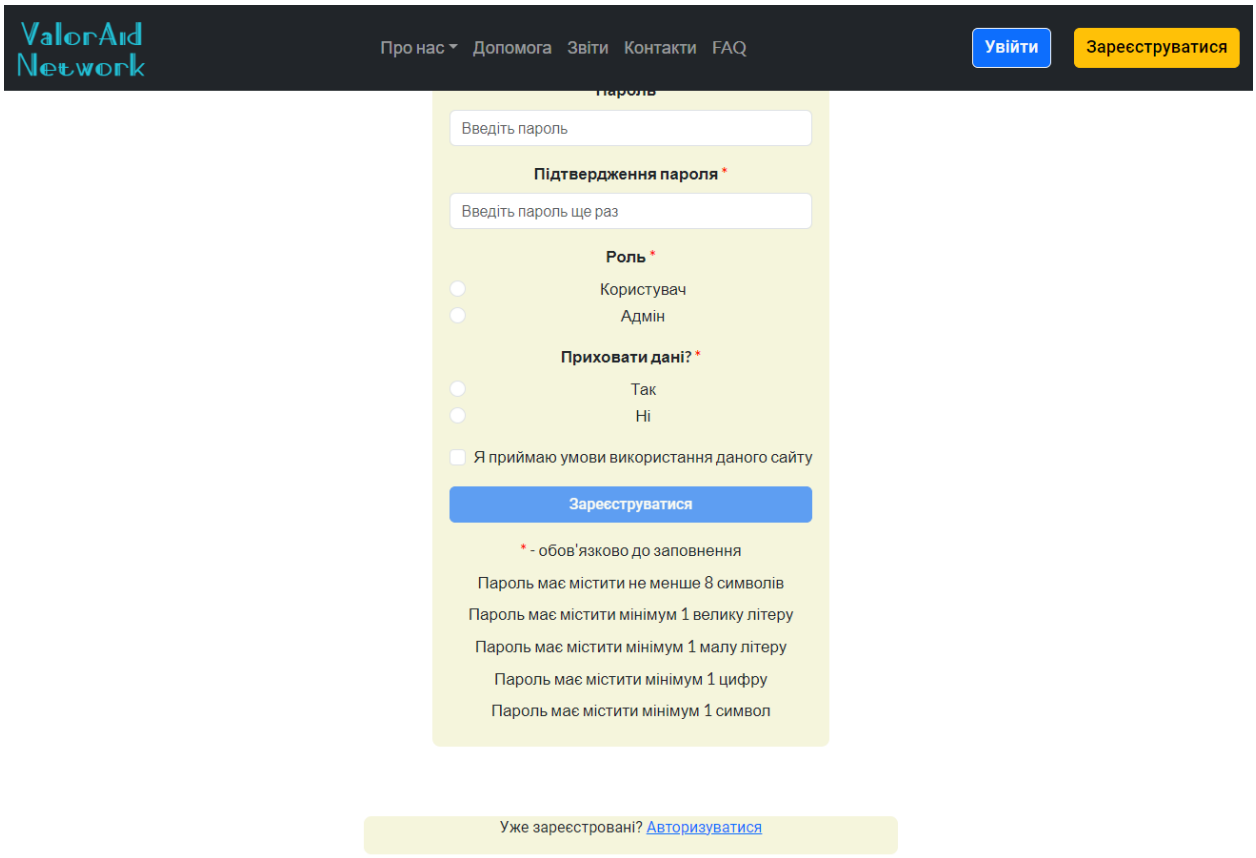

Рисунок 3.20 – Сторінка реєстрації.

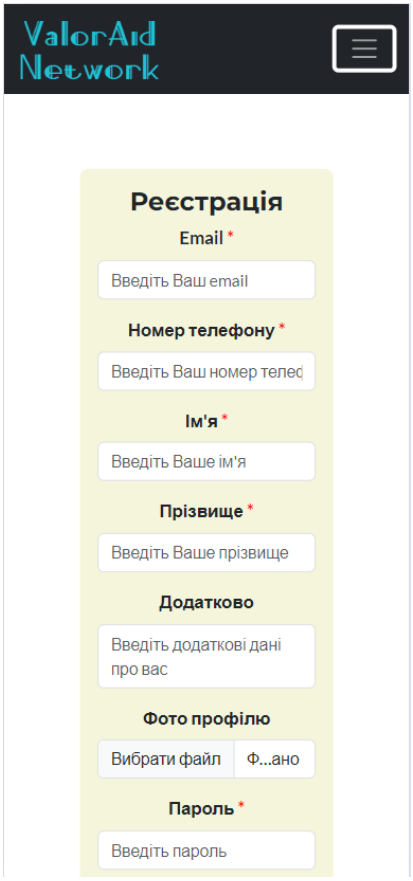

Рисунок 3.21 – Сторінка реєстрації на мобільних пристроях.

Після успішної реєстрації користувач отримує сповіщення про надходження на вказану електронну адресу листа підтвердження, яке дійсне 15 хвилин, і перенаправляється на головну. Результат реєстрації показано на рисунках 3.22-3.23.

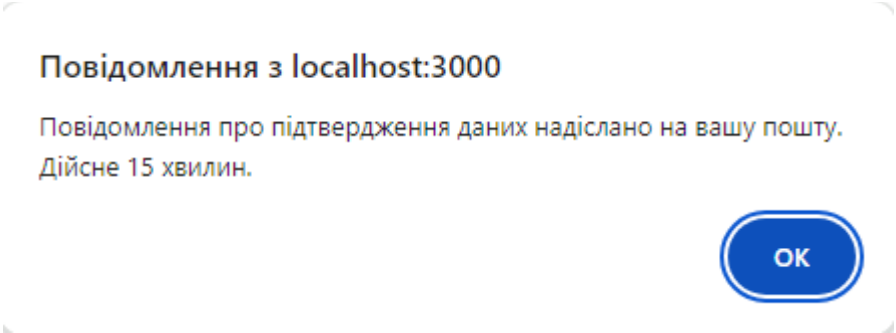

Рисунок 3.22 – Сповіщення про успішну реєстрацію.

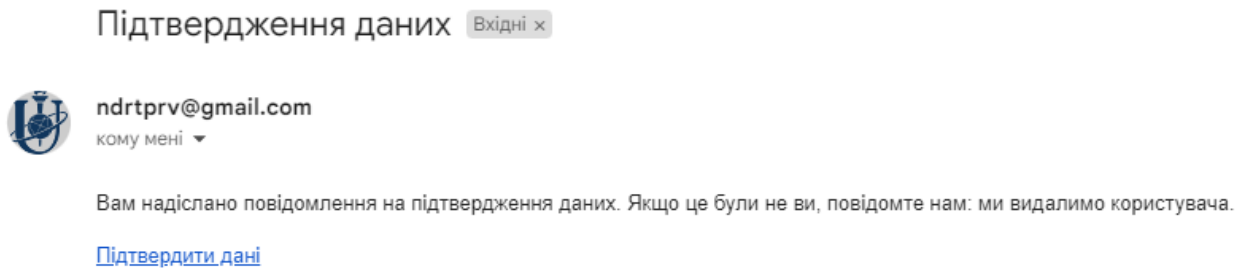

Рисунок 3.23 – Надіслане повідомлення.

Якщо користувач з певних причин не отримав повідомлення, або строк дії сплив, він може зайти на сторінку профілю користувача і перевідправити повідомлення. Процес підтвердження email наведено на рисунках 3.24-3.27.

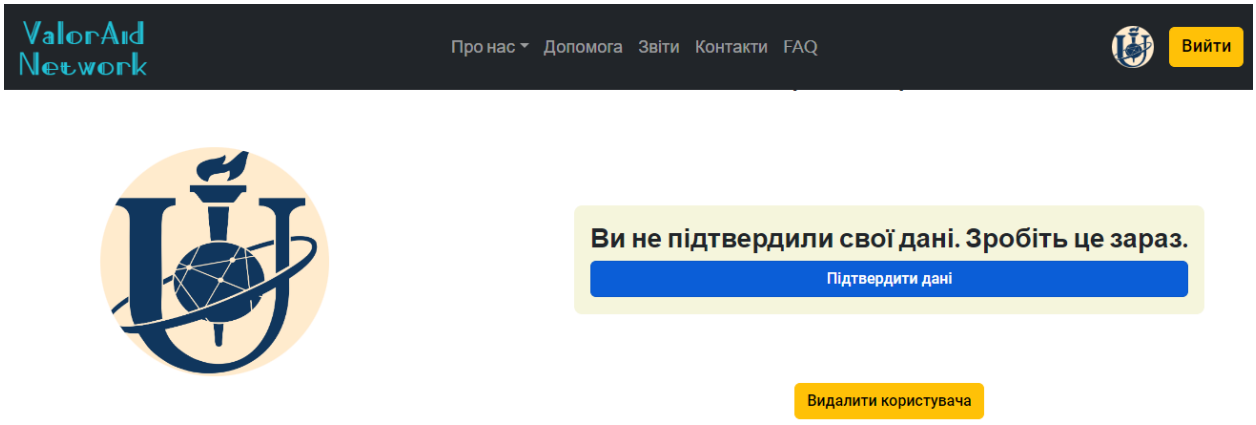

Рисунок 3.24 – Сторінка профілю користувача, який не підтвердив власної пошти.

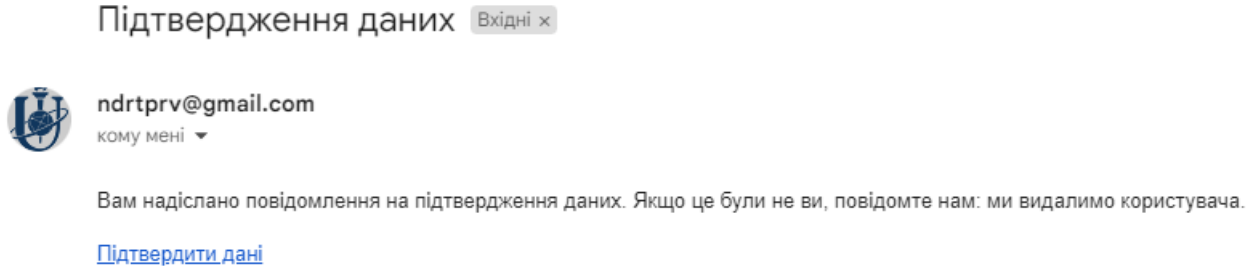

Рисунок 3.25 – Перевідправлене повідомлення.

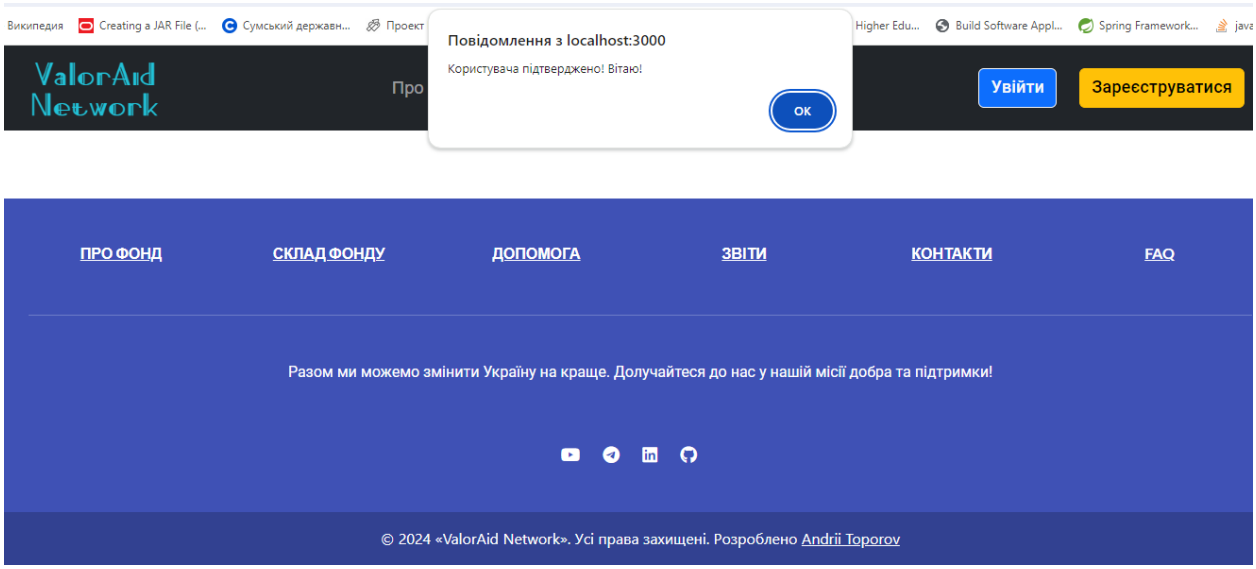

Рисунок 3.26 – Після підтвердження пошти.

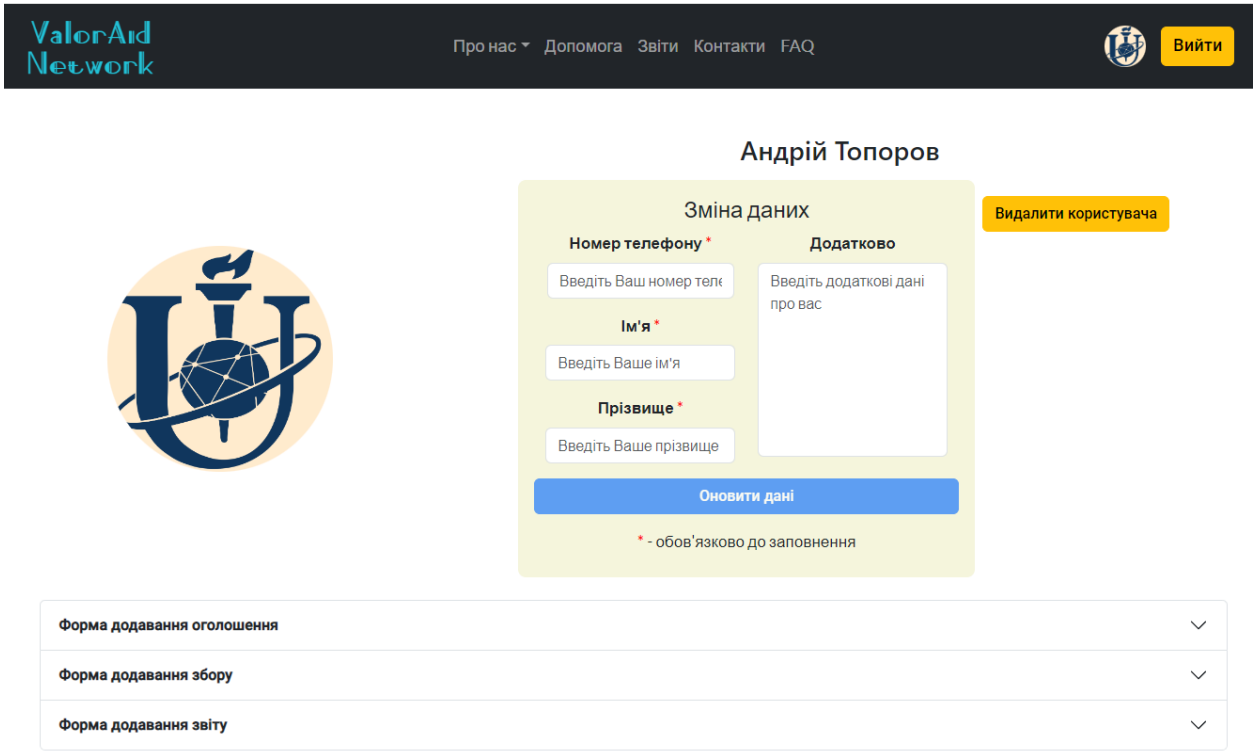

Рисунок 3.27 – Профіль користувача після підтвердження пошти.

3.4.4 Вихід з облікового запису

Для виходу з облікового запису користувача в навігаційній панелі наявна кнопка «Вийти», після натиснення якої вебзастосунок перенаправляє користувача на головну. Вигляд кнопки наявний на рисунку 3.28.

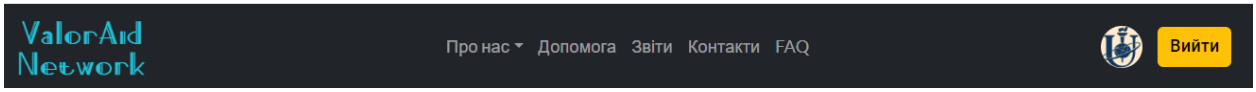

Рисунок 3.28 – Кнопка виходу з облікового запису.

3.4.5 Авторизація

Як і на сторінці реєстрації, форма є основою сторінки авторизації. Але на відміну від реєстрації, для авторизації необхідно ввести електронну адресу та пароль. Вигляд сторінки наведені на рисунках 3.29-3.30.

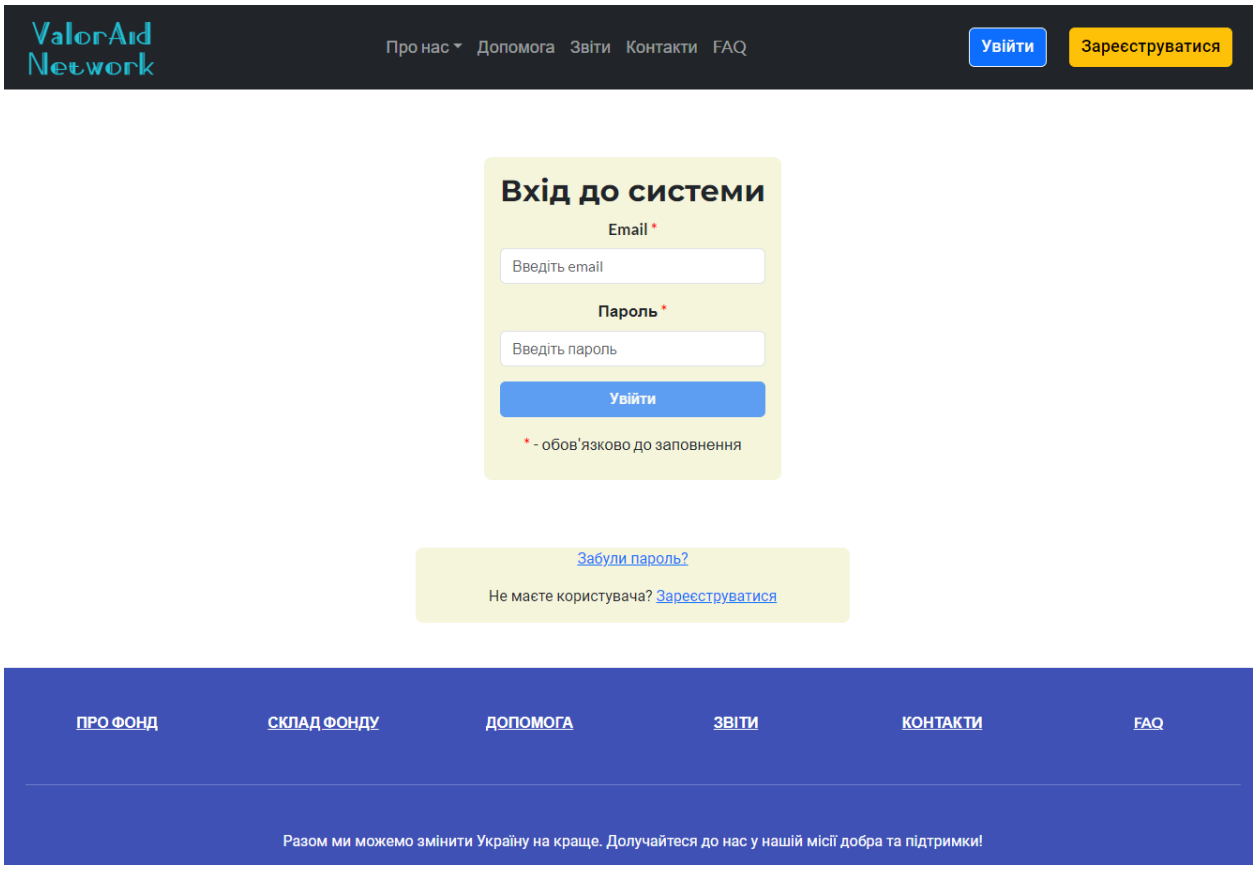

Рисунок 3.29 – Сторінка авторизації.

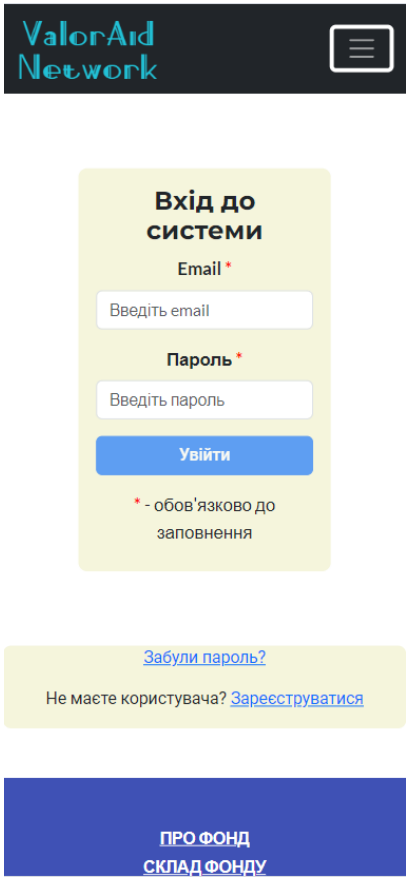

Рисунок 3.30 – Сторінка авторизації для мобільних пристроїв.

Якщо користувача, який ввів email та пароль, знайдено, і пароль є правильним, то його перенаправляють на сторінку профілю користувача.

3.4.6 Сторінка «Забули пароль?» та скидання паролю

Якщо користувач випадково забув власний пароль, він може в будь-який момент його змінити, зайшовши на сторінку авторизації та натиснувши на посилання «Забули пароль?», сторінка якого зображена на рисунках 3.31-3.32.

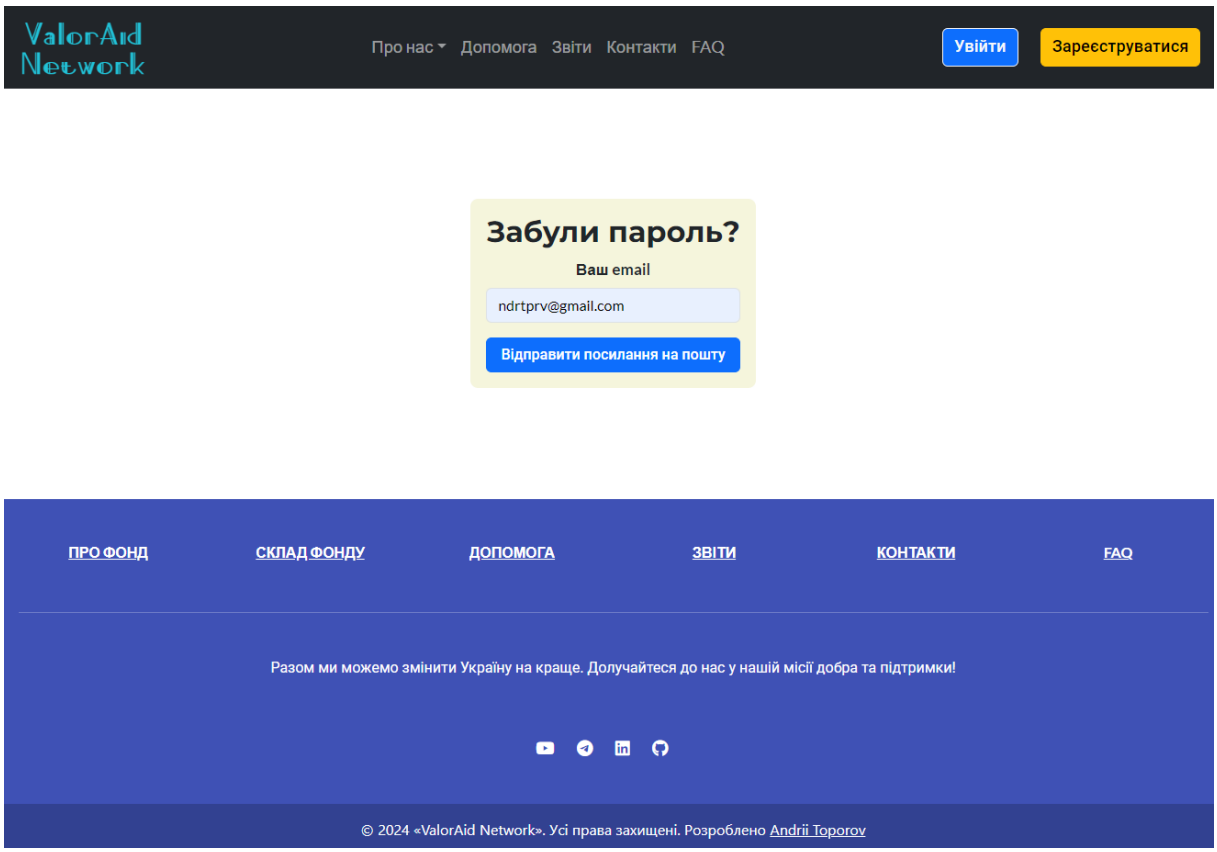

Рисунок 3.31 – Сторінка «Забули пароль?».

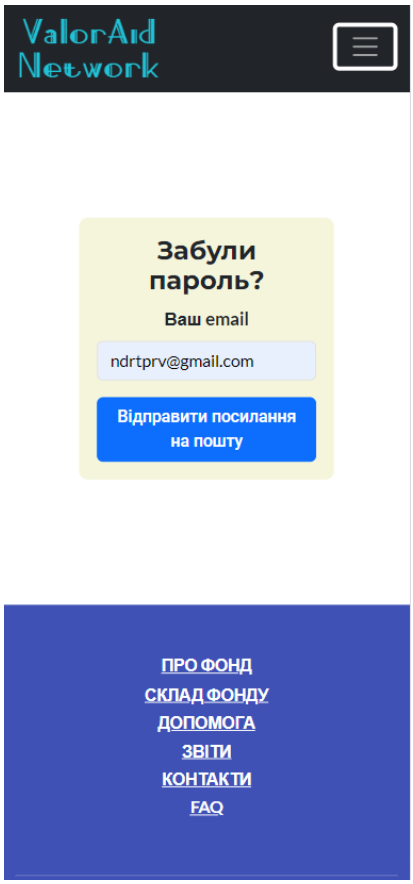

Рисунок 3.32 – Сторінка «Забули пароль?» на мобільних пристроях.

Після надсилання над формою з’являється сповіщення про те, що посилання на сторінку скидання паролю надіслане. Воно, як і посилання на підтвердження пошти, дійсне 15 хвилин. Приклад роботи наведено на рисунках 3.33-3.34.

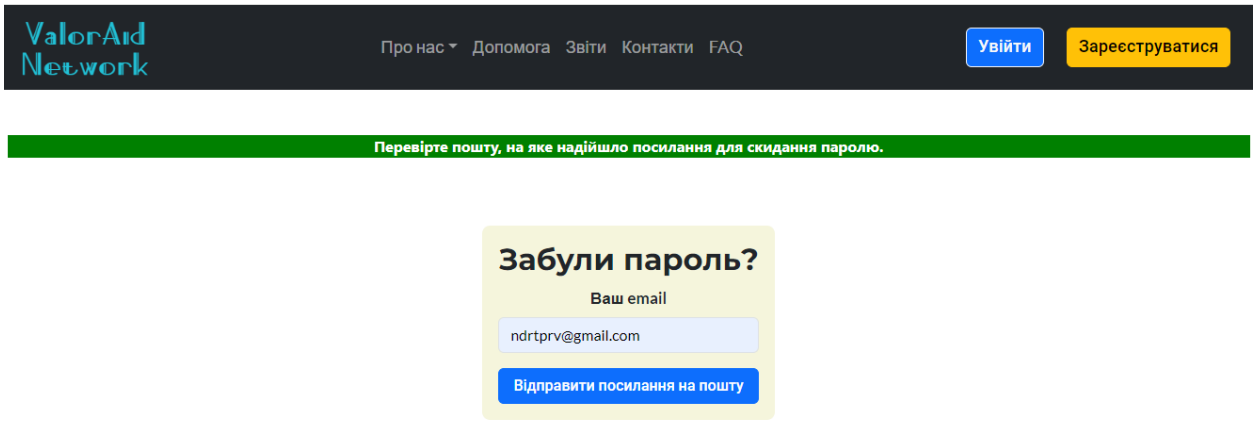

Рисунок 3.33 – Після надсилання повідомлення про скидання паролю.

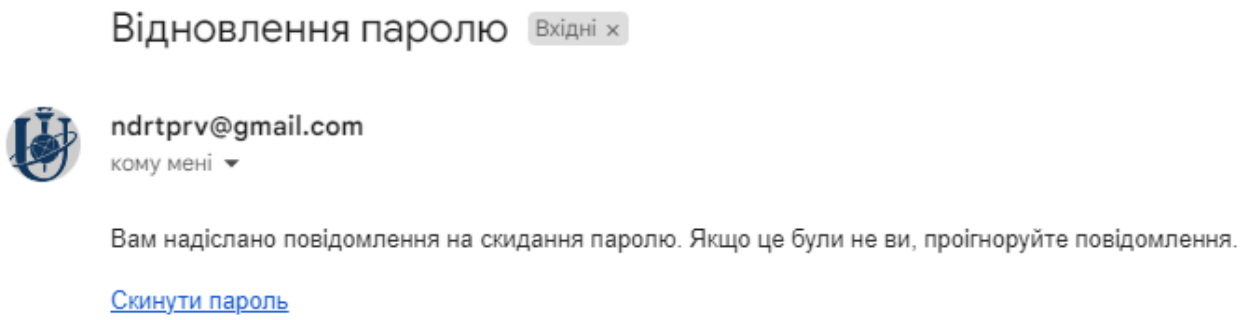

Рисунок 3.34 – Повідомлення.

Після переходу за посиланням користувач опиняється на сторінці скидання паролю. Воно містить форму, яке складається з полів для паролю та підтвердження паролю. Вигляд сторінки можна побачити на рисунках 3.35-3.36.

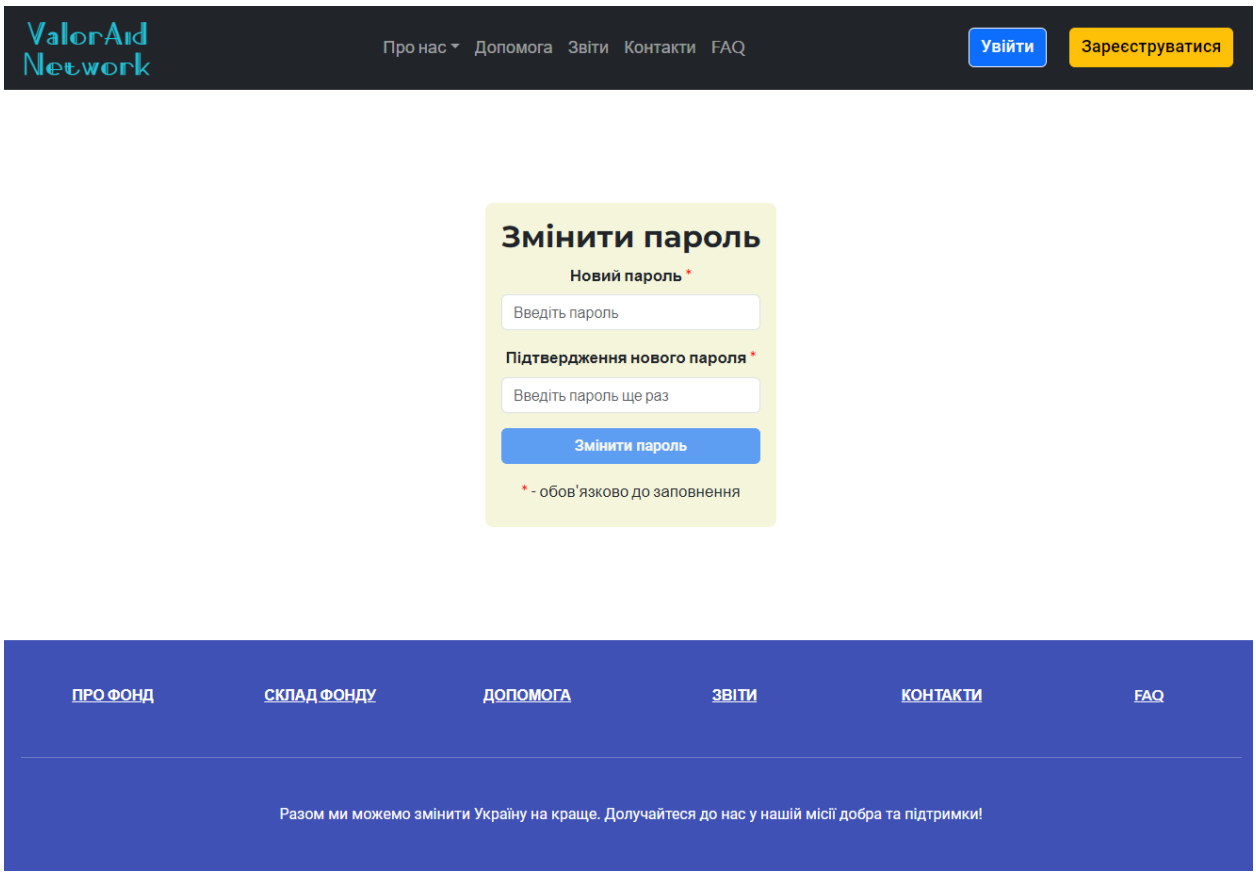

Рисунок 3.35 – Сторінка скидання паролю.

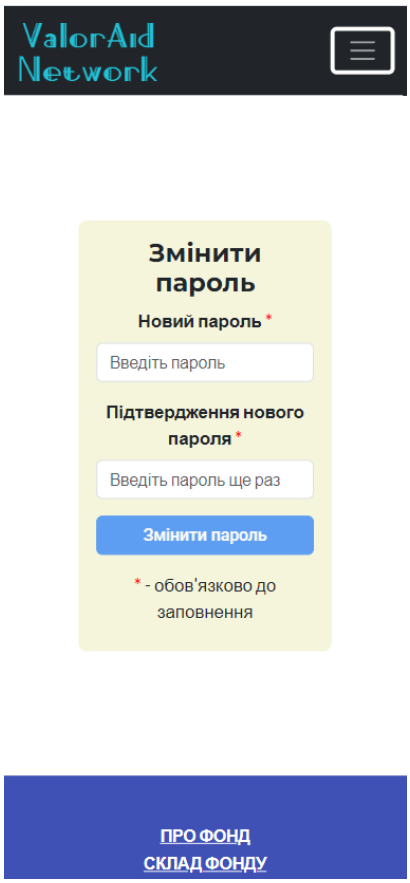

Рисунок 3.36 – Сторінка скидання паролю на мобільних пристроях.

3.4.7 Зміна даних користувача

Після того, як користувач підтвердив власну пошту, він може змінювати дані власного облікового запису у власному профілі, вигляд якого можна побачити на рисунках 3.37-3.38.

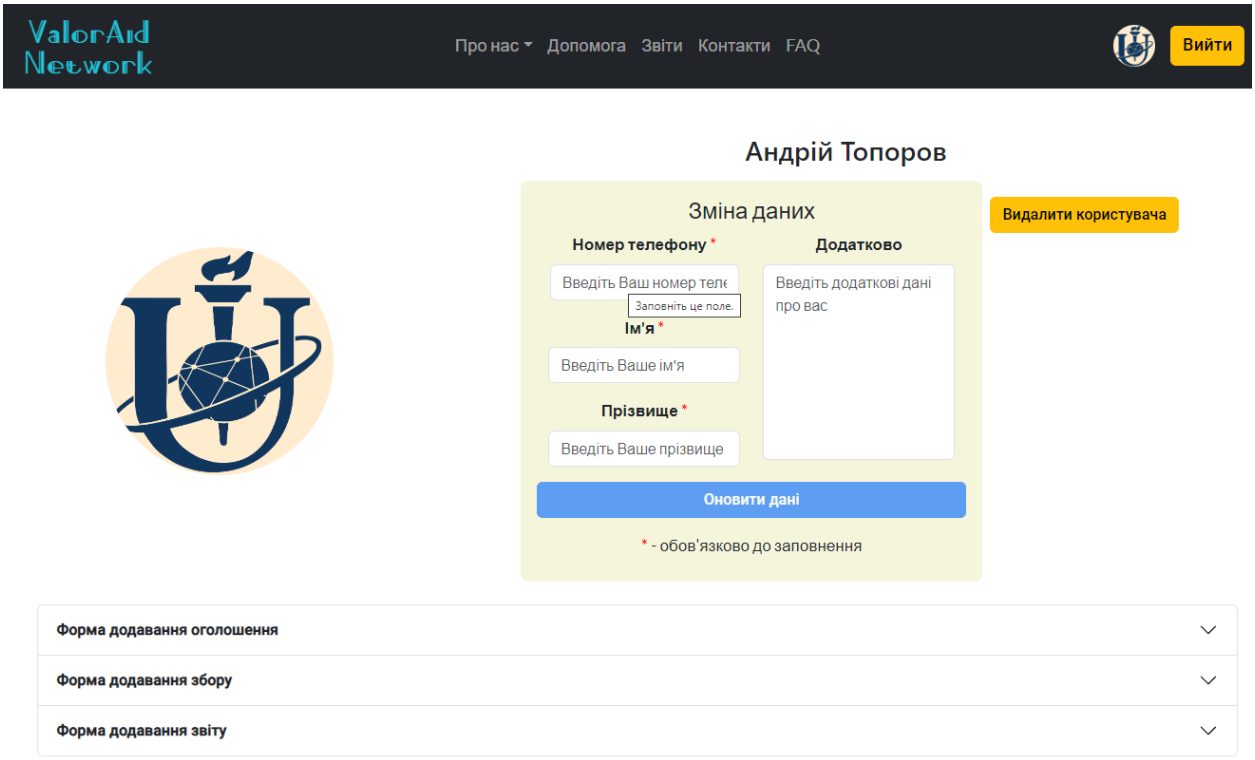

Рисунок 3.37 – Профіль користувача.

Рисунок 3.38 – Профіль користувача на мобільних додатках.

Кнопка оновлення даних буде відключеною, якщо хоча б одне з необхідних полів є пустим, або коли усі введені дані збігаються з даними в БД. Припустимо, потрібно змінити номер телефону та прізвище. Для цього потрібно ввести дані в необхідні поля, при цьому змінивши значення полів номеру телефону та прізвища. При натисканні кнопки «Оновити дані» з’являється сповіщення, яке каже про успіх виконаної операції (рисунок 3.39). Також можна побачити в профілі, що прізвище змінилося (рисунок 3.40).

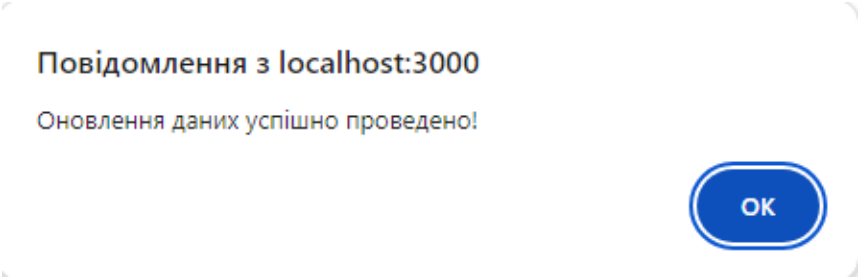

Рисунок 3.39 – Сповіщення про оновлення даних.

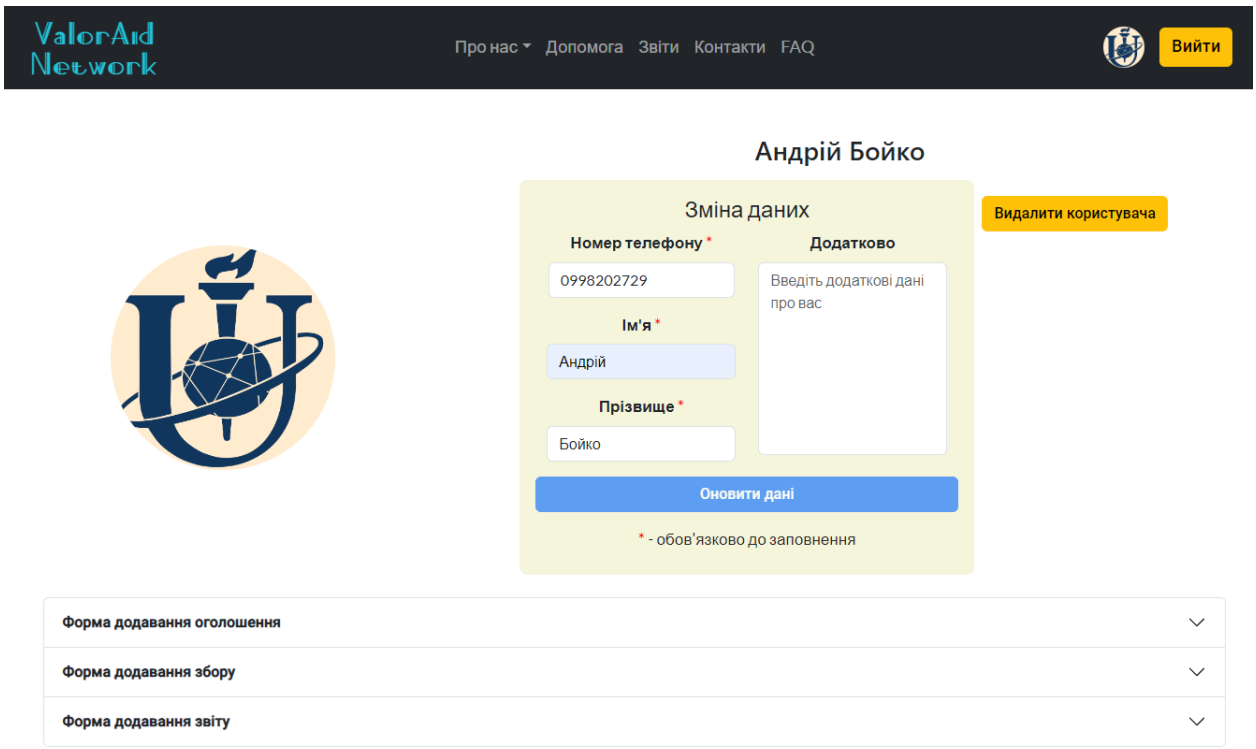

Рисунок 3.40 – Після оновлення даних.

3.4.8 Видалення користувача

Для видалення користувача в профілі існує кнопка «Видалити користувача», яка активна навіть якщо користувач не підтвердив власної пошти. Після натиснення на неї з’являється модальне вікно (рисунок 3.41), яке питає про підтвердження виконання дії. Якщо її підтверджено, то з’являється сповіщення про успіх виконання операції (рисунок 3.42), і вебзастосунок перенаправляє користувача на головну.

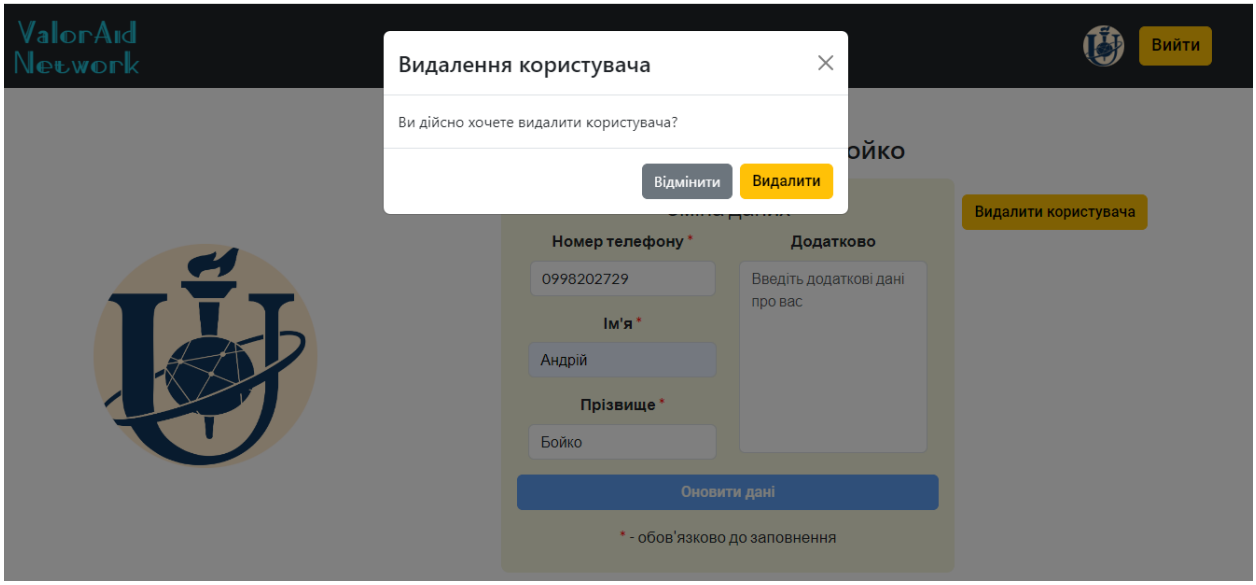

Рисунок 3.41 – Модальне вікно про підтвердження видалення.

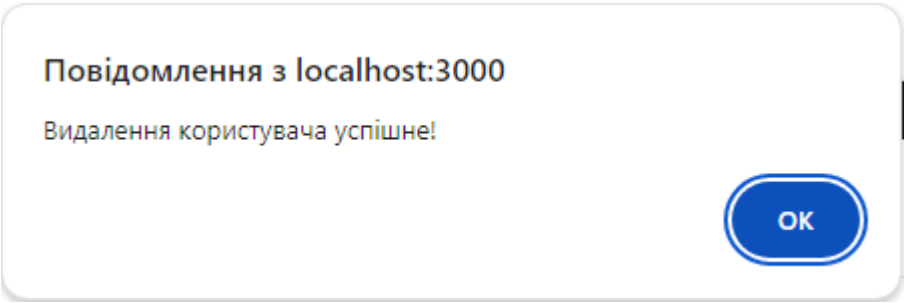

Рисунок 3.42 – Сповіщення про видалення користувача.

3.4.9 Додавання оголошення про надання допомоги

У профілі користувача є можливість додавати оголошення. Для цього існує форма, прихована під компонентом типу Accordion, вигляд якого бачимо на рисунках 3.43-3.44. Як можна побачити, форма має 4 обов’язкові елементи: вибір типу та виду допомоги, її опис та форму вибору фото. При успішному додаванні оголошення з’являється сповіщення про успіх виконання операції (рисунок 3.45).

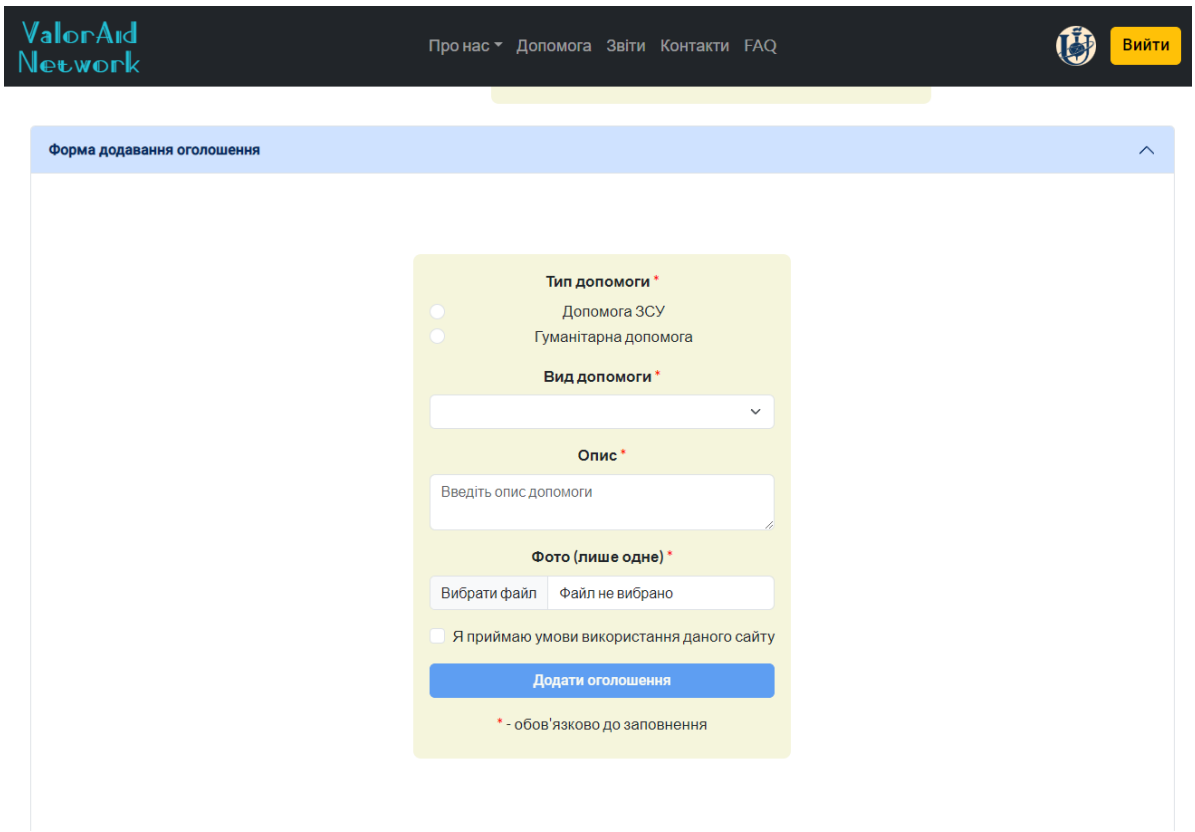

Рисунок 3.43 – Форма додавання оголошення

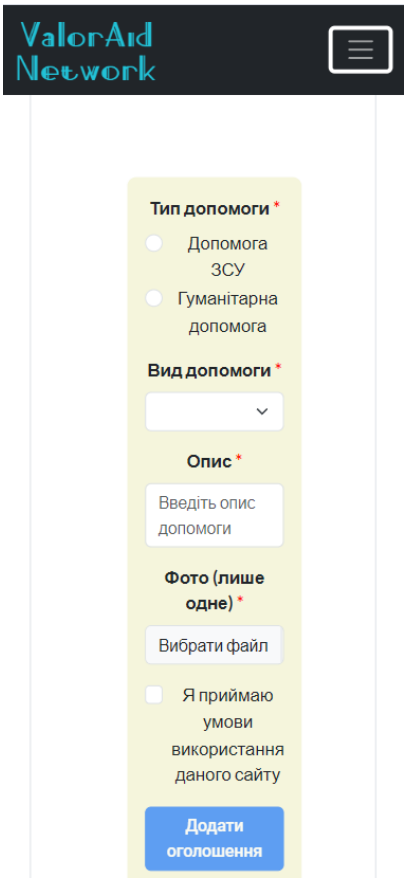

Рисунок 3.44 – Форма додавання оголошення на мобільних девайсах.

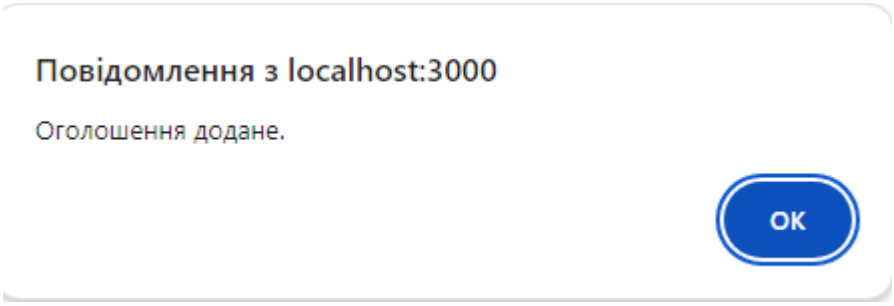

Рисунок 3.45 – Сповіщення про успішне додавання.

3.4.10 Подальша робота

Так як сайт планувався досить великим у майбутньому планується виконати наступні роботи:

1. Редагування, відображення та видалення оголошень з боку користувача та адміністратора.
2. Додавання, редагування зборів з боку користувача, відображення їх на екрані, підтвердження, відхилення та видалення зборів з боку адміністратора.
3. Додавання, відображення та редагування звітів.
4. Створення панелі адміністратора.
5. Додавання платіжних форм для пожертв.
6. Вдосконалення зворотнього зв’язку у вигляді чатів.
7. Додавання локалізації вебзастосунку.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було виконано такі завдання:

1. Визначено актуальність теми роботи.
2. Проведено аналіз існуючих аналогів.
3. Сформовано постановку задачі.
4. Спроєктовано інформаційну модель шляхом побудови функціональної діаграми, діаграми послідовностей, діаграми варіантів використання.
5. Побудовано діаграму відношення сутностей для бази даних.
6. Програмно реалізовано прототип клієнтської та серверної частин вебзастосунку.
7. Проаналізовано отримані результати.

У результаті отримано прототип клієнтської та серверної частини вебзастосунку благодійного фонду, адаптований під екрани мобільних пристроїв, яка надає змогу користувачу можливість реєструватися, приховати чутливі дані задля власної безпеки, підтверджувати власні дані, авторизуватися, змінювати пароль та інші дані, видаляти користувача, та додавати оголошення про надання допомоги.

Надалі планується:

1. Редагування, відображення та видалення оголошень з боку користувача та адміністратора.
2. Додавання, редагування зборів з боку користувача, відображення їх на екрані, підтвердження, відхилення та видалення зборів з боку адміністратора.
3. Додавання, відображення та редагування звітів.
4. Створення панелі адміністратора.
5. Додавання платіжних форм для пожертв.
6. Вдосконалення зворотнього зв’язку у вигляді чатів.
7. Додавання локалізації вебзастосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналіз українського волонтерства на основі методології нових соціальних рухів. *Національний інститут стратегічних досліджень*. URL: https://niss.gov.ua/news/komentari-ekspertiv/analiz-ukrayinskoho-volonterstva-na-osnovi-metodolohiyi-novykh-sotsialnykh (date of access: 28.06.2023).

2. Кількість громадян, залучених до благодійності й волонтерства, зросла вдвічі. *Worldwide online and smartphone surveys | Gradus*. URL: https://gradus.app/uk/open-reports/number-people-involved-charity-and-volunteering-has-doubled-over-past-year/ (дата звернення: 28.06.2023).

3. Українці ще більше донатять на тлі втоми від невизначеної тривалості війни. *Worldwide online and smartphone surveys | Gradus*. URL: https://gradus.app/uk/open-reports/ukrainians-donate-more-amid-fatigue-uncertain-duration-war/ (дата звернення: 24.05.2024).

4. Contributor T. What is Web Application (Web Apps) and its Benefits. *Software Quality*. URL: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app (date of access: 01.07.2023).

5. Благодійний фонд "Повернись живим" – допомога ЗСУ. *savelife.in.ua*. URL: https://savelife.in.ua/ (дата звернення: 01.07.2023).

6. Фонд Сергія Притули. *Фонд Сергія Притули*. URL: https://prytulafoundation.org/ (дата звернення: 01.07.2023).

7. Blagomay | Головна. *Blagomay | Головна*. URL: https://charitymay.com/uk/ (дата звернення: 01.07.2023).

8. Благодійний фонд допомоги ЗСУ – Захист майбутнього. *Захист майбутнього*. URL: https://maibutniefund.org/ (дата звернення: 01.07.2023).

9. Дія – Державні послуги онлайн. *Державні послуги онлайн | Дія*. URL: https://diia.gov.ua/ (дата звернення: 03.07.2023).

10. Методологія idef0. *StudFiles*. URL: https://studfile.net/preview/5203249/page:2/ (дата звернення: 15.07.2023).

11. Maxym Z. Діаграма послідовності (Sequence Diagrams). *Maxym*

Zosym. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 15.07.2023).

12. Інструкція, як будувати UML-діаграми | DOU. *Інструкція, як будувати UML-діаграми | DOU*. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 15.07.2023).

13. El_Rahman S. A., Al-Twaim B. A. Development of Quality Assurance System for Academic Programs and Courses Reports. *International Journal of Modern Education and Computer Science*. 2015. Vol. 7, no. 6. P. 30–36. URL: https://doi.org/10.5815/ijmecs.2015.06.05 (date of access: 24.05.2024).

14. Uzayr S. b. Mastering React: A Beginner's Guide. Taylor & Francis Group, 2022.

15. Get started with Bootstrap. *Bootstrap – The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (date of access: 11.07.2023).

16. Buckler C. Node. js: Novice to Ninja. SitePoint Pty, Limited, 2022.

17. PostgreSQL Tutorial. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/postgresql/index.php (date of access: 11.07.2023).

18. React Router: Declarative Routing for React. *ReactRouterWebsite*. URL: https://v5.reactrouter.com/web/guides/quick-start (date of access: 24.05.2024).

19. React Bootstrap | React Bootstrap. *React Bootstrap | React Bootstrap*. URL: https://react-bootstrap.netlify.app/ (date of access: 24.05.2024).

20. React Avatar component – Material UI. *MUI: The React component library you always wanted*. URL: https://mui.com/material-ui/react-avatar/ (date of access: 24.05.2024).

21. Починаючи | Axios Docs. *Axios*. URL: https://axios-http.com/uk/docs/intro (дата звернення: 24.05.2024).

22. Cross-Origin Resource Sharing (CORS) – HTTP | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (date of access: 24.05.2024).

23. bcrypt. *npm*. URL: https://www.npmjs.com/package/bcrypt (date of

access: 24.05.2024).

24. Getting Started | Sequelize. *Sequelize | Feature-rich ORM for modern TypeScript & JavaScript*. URL: https://sequelize.org/docs/v6/getting-started/ (date of access: 24.05.2024).

25. dotenv. *npm*. URL: https://www.npmjs.com/package/dotenv (date of access: 24.05.2024).

26. express-validator | express-validator. *express-validator*. URL: https://express-validator.github.io/docs (date of access: 24.05.2024).

27. jsonwebtoken. *npm*. URL: https://www.npmjs.com/package/jsonwebtoken (date of access: 24.05.2024).

28. cookie-parser. *npm*. URL: https://www.npmjs.com/package/cookie-parser (date of access: 24.05.2024).

29. Nodemailer :: Nodemailer. *Nodemailer :: Nodemailer*. URL: https://www.nodemailer.com/ (date of access: 24.05.2024).

30. multer. *npm*. URL: https://www.npmjs.com/package/multer (date of access: 24.05.2024).

31. Microsoft. Documentation for Visual Studio Code. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs (date of access: 16.07.2023).

ДОДАТОК А

Клієнтська частина вебзастосунку

Решта коду розміщена на GitHub за посиланням

https://github.com/ndrtprv/bachelor_work_final

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min.js';

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

App.js

```
import React from 'react';
import './App.css';
import { BrowserRouter } from 'react-router-dom';
import AppRouter from './components/app_router/AppRouter';
import Header from './components/header/Header';
import Footer from './components/footer/Footer';
import { paths_data1, paths_data2, media_paths } from './utils/constants';

function App() {

  return (
      <div className='App'>
        <BrowserRouter>
          <Header data1={paths_data1} data2={paths_data2} />
          <AppRouter />
          <Footer data1={paths_data1} data2={paths_data2} data3={media_paths} />
        </BrowserRouter>
      </div>
  );
}

export default App;
```

Header.js

```
import React from 'react';
```

```
import './Header.css';
import brand from '../../resources/brand.png';
import NavigationPanel from './nav_panel/NavigationPanel';
import avatar from '../../resources/people.png';

function Header(props) {

    return (
        <header className="sticky-top">
            <NavigationPanel data1={props.data1} data2={props.data2} brand={brand}
avatar={avatar} />
        </header>
    );
}

export default Header;
```

NavigationPanel.js:

```
import { NavLink, useNavigate } from 'react-router-dom';
import { LANDING_ROUTE, USER_ROUTE, LOGIN_ROUTE, SIGNUP_ROUTE, ADMIN_ROUTE } from
'../../../utils/constants';
import { Container, Nav, Navbar, NavDropdown, Button } from 'react-bootstrap';
import axios from 'axios';
import Avatar from 'react-avatar';
import { useEffect, useState } from 'react';

function NavigationPanel(props) {

    const navigate = useNavigate();
    const [isLoggedIn, setIsLoggedIn] = useState(false);
    const [isAdmin, setIsAdmin] = useState(false);
    const [avatar, setAvatar] = useState(props.avatar);

    axios.defaults.withCredentials = true;
    useEffect(() => {
        axios.get(process.env.REACT_APP_API_URL + 'user/nav')
        .then(res => {
            if (res.data.status) {
                setIsLoggedIn(res.data.status);

                if (res.data.isAdmin) {
                    setIsAdmin(res.data.isAdmin)
                }

                if (res.data.portrait !== null) {
                    setAvatar(res.data.portrait);
                }
            } else {
                setIsLoggedIn(false);
```

```
                setIsAdmin(false);
            }
        }).catch(err => {
            console.log(err.message);
        })
    });

    const logOut = () => {
        axios.get('http://localhost:3003/user/logout')
        .then(res => {
            if (res.data.status) {
                navigate(LANDING_ROUTE);
            }
        }).catch(e => {
            console.log(e);
        })
    }

    return (
        <Navbar collapseOnSelect expand="lg" className="navbar-dark bg-dark" aria-
label="navbar">
            <Container className="container-fluid">
                <NavLink to={LANDING_ROUTE} className="navbar-brand"><img src={props.brand}
alt="ValorAid Network" className='mybrand' /></NavLink>
                <Navbar.Toggle aria-controls="responsive-navbar-nav" />

                <Navbar.Collapse id="responsive-navbar-nav">
                    <Nav className="navbar-nav me-auto mx-auto">
                        <NavDropdown title="Про нас" className="nav-item dropdown">
                            {Object.entries(props.data1).map(([path, label]) =>
                                <NavLink key={path} to={path} className="dropdown-item
lat">{label}</NavLink>
                            )}
                        </NavDropdown>
                        {Object.entries(props.data2).map(([path, label]) =>
                            <NavLink key={path} to={path} className="nav-link
lat">{label}</NavLink>
                        )}
                    </Nav>
                    {
                        isLoggedIn ?
                        <Nav className="nav navbar-right">
                            <NavLink to={USER_ROUTE} className="nav-link lat">
                                <Avatar alt="Профіль" src={avatar !== props.avatar ?
`data:${avatar.contentType};base64,${avatar.data}` : props.avatar} size="2.4em" />
                            </NavLink>
                            {
                                isAdmin ?
                                <NavLink to={ADMIN_ROUTE} className="nav-link lat">
                                    <Button type="button" className="btn btn-primary my-2 rob-
btn" >Адмін панель</Button>
```

```
                                </NavLink>
                                :
                                <></>
                            }
                            <NavLink to={LANDING_ROUTE} className="nav-link lat">
                                <Button type="button" className="btn btn-warning my-2 rob-btn"
onClick={logOut}>Вийти</Button>
                            </NavLink>
                        </Nav>
                        :
                        <Nav className="nav navbar-right">
                            <NavLink to={LOGIN_ROUTE} className="nav-link lat">
                                <Button type="button" className="btn btn-outline-light me-2
my-2 rob-btn">Увійти</Button>
                            </NavLink>
                            <NavLink to={SIGNUP_ROUTE} className="nav-link lat">
                                <Button type="button" className="btn btn-warning my-2 rob-
btn">Зареєструватися</Button>
                            </NavLink>
                        </Nav>
                    }
                </Navbar.Collapse>
            </Container>
        </Navbar>
    );
}

export default NavigationPanel;
```

Footer.js

```
import React from 'react';
import './Footer.css';
import { Link } from 'react-router-dom';
import { Container } from 'react-bootstrap';

function Footer(props) {

  var cop_text = "© " + new Date().getFullYear() + " «ValorAid Network». Усі права захищені.";
  const merged_data = {...props.data1, ...props.data2};

  return (
    <footer className="footer text-center text-white foot-bottom">
      <Container className="container">
        <section className="mt-5">
          <div className="row text-center d-flex justify-content-center pt-5">
            {Object.entries(merged_data).map(([path, label]) =>
              <div className="col-md-2">
                <h6 className="text-uppercase font-weight-bold lat">
                  <Link key={path} to={path} className="text-white">{label}</Link>
```

```
                </h6>
              </div>
            )}
          </div>
        </section>
        <hr className="my-5" />
        <section className="mb-5">
          <div className="row d-flex justify-content-center">
            <div className="col-lg-8">
              <p className='rob'>
                Разом ми можемо змінити Україну на краще. Долучайтеся до нас у нашій місії
добра та підтримки!
              </p>
            </div>
          </div>
        </section>
        <section className="text-center mb-5">
          {Object.entries(props.data3).map(([path, label]) =>
            <Link key={path} to={label} className="text-white me-4">
              <i key={path} className={path}></i>
            </Link>
          )}
        </section>
      </Container>
      <div className="text-center p-3 foot-back">
        {cop_text} Розроблено <Link to="https://github.com/ndrtprv" className="text-
white">Andrii Toporov</Link>
      </div>
    </footer>
  );
}

export default Footer;
```

AppRouter.js

```
import React from 'react';
import {Route, Routes, Navigate} from 'react-router-dom';
import { adminRoutes, userRoutes, publicRoutes } from '../../routes';
import { LANDING_ROUTE } from '../../utils/constants';

function AppRouter() {

  return (
    <Routes>
      {adminRoutes.map(({path, Component}) =>
        <Route key={path} path={path} element={Component} exact />
      )}
      {userRoutes.map(({path, Component}) =>
        <Route key={path} path={path} element={Component} exact />
```

```
      )}
      {publicRoutes.map(({path, Component}) =>
        <Route key={path} path={path} element={Component} exact />
      )}
      <Route path='*' element={<Navigate to={LANDING_ROUTE} />} />
    </Routes>
  )
}

export default AppRouter;
```

Landing.js

```
import React from 'react';
import { Button } from 'react-bootstrap';
import { NavLink } from "react-router-dom";
import './Landing.css';
import emp_avatar from '../../resources/people.png';
import uaf_help from '../../resources/uaf_help.jpg';
import humanitarian from '../../resources/humanitarian.jpg';
import volunteer from '../../resources/volunteer.jpg';
import about_fund from '../../resources/about_fund.jpg';
import fundraise from '../../resources/fundraise.jpg';
import results from '../../resources/results.jpg';
import Carousel from '../../components/carousel/Carousel';
import { ABOUT_ROUTE, FUNDRAISINGS_ROUTE, RESULTS_ROUTE, STAFF_ROUTE } from
'../../utils/constants';
import Feedback from '../../components/feedback/Feedback';

function Landing(props) {

  const headerFeedback = "Зворотній зв'язок";
  const textFeedback = "Якщо вас цікавлять певні питання, зв'яжіться з нами, заповнивши
форму.";
  const classNameContainer = "d-flex flex-column justify-content-center";
  const classNameForm = "p-2";

  return (
    <main>
      <Carousel uaf_help={uaf_help} humanitarian={humanitarian} volunteer={volunteer} />
      <div className="container marketing">
        <div className="row featurette start_elem">
          <div className="col-md-7 txt_field">
            <h2 className="featurette-heading lat-h2">Про благодійний фонд</h2>
            <p className="lead">"ValorAid Network" - це благодійний фонд, який об'єднує
зусилля для надання підтримки на передовій лінії ЗСУ та надання гуманітарної допомоги тим, хто
постраждав від наслідків конфлікту. Наш фонд зобов'язаний підтримувати і вдягати героїв на
```

```
фронті та протягом мирних днів надавати допомогу потребуючим для відновлення життя.</p>
            <p className='rob-btn'>
              <NavLink to={ABOUT_ROUTE} className="nav-link lat">
                <Button type="button" className="btn btn-info" >Докладніше</Button>
              </NavLink>
            </p>
          </div>
          <div className="col-md-5">
            <svg className="bd-placeholder-img bd-placeholder-img-lg featurette-image img-
fluid mx-auto" width="500" height="500" xmlns="http://www.w3.org/2000/svg" role="img" aria-
label="Placeholder: 500x500" preserveAspectRatio="xMidYMid slice" focusable="false"><image
href={about_fund} width="100%" height="100%" /></svg>
          </div>
        </div>
        <hr className="featurette-divider"/>
        <div className="row">
          <h2 className="featurette-heading lat-h2">Склад благодійного фонду</h2>
          {Object.entries(props.members).map(([path, label]) =>
            <div className="col-lg-4" key={path}>
              <svg className="bd-placeholder-img rounded-circle" width="140" height="140"
xmlns="http://www.w3.org/2000/svg" role="img" aria-label="Placeholder: 140x140"
preserveAspectRatio="xMidYMid slice" focusable="false"><title>Placeholder</title><image
href={emp_avatar} width="140" height="140" /></svg>
              <h3 className='lat-h3'>{path}</h3>
              <p className='rob'>{label}</p>
              <p className='rob-btn'>
                <NavLink to={STAFF_ROUTE} className="nav-link lat">
                  <Button type="button" className="btn btn-secondary" >Деталі »</Button>
                </NavLink>
              </p>
            </div>
          )}
        </div>
        <hr className="featurette-divider"/>
        <div className="row featurette">
          <div className="col-md-7 order-md-2 txt_field">
            <h2 className="featurette-heading lat-h2">Збори</h2>
            <p className="lead">Місце, де ви можете долучитися до нашої місії та внести свій
внесок у підтримку ЗСУ та допомогу тим, хто постраждав від війни. Тут ви зможете знайти
різноманітні можливості пожертвувань та долучитися до наших проектів для спільного досягнення
позитивних змін.</p>
            <p className='rob-btn'>
              <NavLink to={FUNDRAISINGS_ROUTE} className="nav-link lat">
                <Button type="button" className="btn btn-info" >Актуальні збори</Button>
              </NavLink>
            </p>
```

```
          </div>
          <div className="col-md-5 order-md-1">
            <svg className="bd-placeholder-img bd-placeholder-img-lg featurette-image img-
fluid mx-auto" width="500" height="500" xmlns="http://www.w3.org/2000/svg" role="img" aria-
label="Placeholder: 500x500" preserveAspectRatio="xMidYMid slice" focusable="false"><image
href={fundraise} width="100%" height="100%" /></svg>
          </div>
        </div>
        <hr className="featurette-divider"/>
        <div className="row featurette">
          <div className="col-md-7 txt_field">
            <h2 className="featurette-heading lat-h2">Звіти</h2>
            <p className="lead">Ми ділимося докладними звітами про використання пожертвувань
та реалізовані проекти. Тут ви зможете дізнатися, як ваша підтримка сприяє покращенню життя
воїнів ЗСУ та тих, хто потерпів від війни, і бачити конкретні результати нашої спільної
діяльності.</p>
            <p className='rob-btn'>
              <NavLink to={RESULTS_ROUTE} className="nav-link lat">
                <Button type="button" className="btn btn-info" >Усі звіти</Button>
              </NavLink>
            </p>
          </div>
          <div className="col-md-5">
            <svg className="bd-placeholder-img bd-placeholder-img-lg featurette-image img-
fluid mx-auto" width="500" height="500" xmlns="http://www.w3.org/2000/svg" role="img" aria-
label="Placeholder: 500x500" preserveAspectRatio="xMidYMid slice" focusable="false"><image
href={results} width="100%" height="100%" /></svg>
          </div>
        </div>
        <hr className="featurette-divider"/>
        <div className="row featurette">
          <Feedback isSmall={false} headerFeedback={headerFeedback}
textFeedback={textFeedback} classNameContainer={classNameContainer}
classNameForm={classNameForm} />
        </div>
      </div>
    </main>
  );
}

export default Landing;
```

Carousel.js

```
import { NavLink } from "react-router-dom";
import { FUNDRAISINGS_ROUTE, SIGNUP_ROUTE } from "../../utils/constants";
```

```
import { Container } from "react-bootstrap";

function Carousel(props) {

    return (
        <div id="myCarousel" className="carousel slide" data-bs-ride="carousel">
            <div className="carousel-indicators">
                <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="0"
className="" aria-label="Slide 1"></button>
                <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="1" aria-
label="Slide 2" className="active" aria-current="true"></button>
                <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="2" aria-
label="Slide 3" className=""></button>
            </div>
            <div className="carousel-inner">
                <div className="carousel-item">
                    <svg className="bd-placeholder-img img-uaf" width="100%" height="100%"
xmlns="http://www.w3.org/2000/svg" aria-hidden="true" preserveAspectRatio="xMidYMid slice"
focusable="false"><image href={props.uaf_help} width="100%" height="100%" /></svg>
                    <Container >
                        <div className="carousel-caption text-start">
                            <h1 className='d-inline-block mb-1 mont'>Допомога ЗСУ</h1>
                            <p className='rob carous'>Підтримка Збройних Сил України є
важливим вкладом у забезпечення національної безпеки та суверенітету, а також наближає
перемогу у війні з агресором.</p>
                            <p className='rob-btn'><NavLink className="btn btn-lg btn-primary"
to={FUNDRAISINGS_ROUTE} >Підтримати ЗСУ</NavLink></p>
                        </div>
                    </Container>
                </div>
                <div className="carousel-item active">
                    <svg className="bd-placeholder-img img-hum" width="100%" height="100%"
xmlns="http://www.w3.org/2000/svg" aria-hidden="true" preserveAspectRatio="xMidYMid slice"
focusable="false"><image href={props.humanitarian} width="100%" height="100%" /></svg>
                    <Container >
                        <div className="carousel-caption">
                            <h1 className='d-inline-block mb-1 mont'>Гуманітарна допомога</h1>
                            <p className='rob carous'>Наша спільна діяльність спрямована на
полегшення страждань та покращення умов тих, хто потребує допомоги, у тому числі постраждалим
від війни.</p>
                            <p className='rob-btn'><NavLink className="btn btn-lg btn-primary"
to={FUNDRAISINGS_ROUTE} >Пожертвувати</NavLink></p>
                        </div>
                    </Container>
                </div>
                <div className="carousel-item">
```

```
                    <svg className="bd-placeholder-img img-vol" width="100%" height="100%"
xmlns="http://www.w3.org/2000/svg" aria-hidden="true" preserveAspectRatio="xMidYMid slice"
focusable="false"><image href={props.volunteer} width="100%" height="100%" /></svg>
                    <Container >
                        <div className="carousel-caption text-end">
                            <h1 className='d-inline-block mb-1 mont'>Стати волонтером</h1>
                            <p className='rob carous'>Бажаєш також допомагати постраждалим від
війни або Збройним Силам? Тоді долучайся до нашої команди волонтерів! Заповнюй анкету і стань
одним з нас.</p>
                            <p className='rob-btn'><NavLink className="btn btn-lg btn-primary"
to={SIGNUP_ROUTE} >Приєднатися зараз</NavLink></p>
                        </div>
                    </Container>
                </div>
            </div>
            <button className="carousel-control-prev" type="button" data-bs-
target="#myCarousel" data-bs-slide="prev">
                <span className="carousel-control-prev-icon" aria-hidden="true"></span>
                <span className="visually-hidden">Попередній</span>
            </button>
            <button className="carousel-control-next" type="button" data-bs-
target="#myCarousel" data-bs-slide="next">
                <span className="carousel-control-next-icon" aria-hidden="true"></span>
                <span className="visually-hidden">Наступний</span>
            </button>
        </div>
    );
}

export default Carousel;
```

About.js

```
import { Container } from 'react-bootstrap';

function About() {
    return (
        <main>
            <Container className="mt-3">
                <h2>Про благодійний фонд</h2>
            </Container>
            <Container>
                <h3 style={{textAlign: 'left'}}>Ласкаво просимо до ValorAid Network!</h3>
                <p style={{textAlign: 'left'}}>Ми є благодійним фондом, створеним з метою
підтримки та допомоги Силам оборони України та
                    надання гуманітарної допомоги тим, хто постраждав від наслідків війни.
```

```
Наша місія полягає
                    в об'єднанні зусиль задля підтримки тих, хто бореться за свободу,
незалежність та мир у нашій країні.
                </p>

                <h4 style={{textAlign: 'left'}}>Наші цілі та завдання</h4>
                <h5 style={{textAlign: 'left'}}>Допомога Силам оборони України</h5>
                <p style={{textAlign: 'left'}}>ValorAid Network активно підтримує українських
військових, які захищають наші кордони та незалежність.
                    Ми забезпечуємо їх необхідним обладнанням, медичним спорядженням,
транспортними засобами
                    та іншими важливими ресурсами. Разом ми зміцнюємо нашу обороноздатність та
забезпечуємо гідний захист наших громадян.
                </p>

                <h5 style={{textAlign: 'left'}}>Гуманітарна допомога</h5>
                <p style={{textAlign: 'left'}}>Ми не забуваємо і про мирних жителів, які
постраждали від бойових дій. Наша гуманітарна допомога охоплює
                    постачання продовольства, медикаментів, одягу та інших необхідних речей
для людей, що опинилися у скрутному становищі.
                    Ми прагнемо повернути надію та впевненість тим, хто втратив усе через
війну.
                </p >

                <h4 style={{textAlign: 'left'}}>Як ви можете допомогти</h4>
                <h5 style={{textAlign: 'left'}}>Пожертвування</h5>
                <p style={{textAlign: 'left'}}>Ваша фінансова підтримка є надзвичайно важливою
для нас. Кожна пожертва допомагає нам забезпечити необхідну допомогу тим,
                    хто її потребує найбільше. Ви можете зробити свій внесок через наш вебсайт
або звернутися до нас для отримання додаткової інформації.
                </p>

                <h5 style={{textAlign: 'left'}}>Волонтерство</h5>
                <p style={{textAlign: 'left'}}>Ми завжди шукаємо волонтерів, готових допомогти
у наших проектах. Якщо ви маєте бажання та можливість долучитися до нашої команди,
                    будь ласка, зареєструйтеся, тим самим надіславши заявку. Ваша підтримка на
місцях може змінити життя багатьох людей.
                </p>

                <h5 style={{textAlign: 'left'}}>Поширення інформації</h5>
                <p style={{textAlign: 'left'}}>Допоможіть нам поширити інформацію про нашу
діяльність. Розкажіть своїм друзям, родичам та колегам про ValorAid Network.
                    Разом ми можемо досягти більшого!
                </p>
            </Container>
        </main>
```

```
    );
}

export default About;
```

Contacts.js

```
import './Contacts.css';
import Feedback from '../../components/feedback/Feedback';
import OtherContacts from '../../components/other_contacts/OtherContacts';

function Contacts() {

    const classNameForm = "m-5 p-2";

    return (
        <main>
            <div className='d-flex justify-content-center contact-page'>
                <OtherContacts />
                <Feedback isSmall={true} classNameForm={classNameForm} />
            </div>
        </main>
    );
}

export default Contacts;
```

OtherContacts.js

```
import { Container } from "react-bootstrap";

function OtherContacts() {
    return (
        <Container className="pl-3 pr-3 d-flex flex-column justify-content-center contact-
data">
            <div>
                <h2>Контактні дані</h2>
            </div>
            <div>
                <p>Тут ви маєте можливість ознайомитися з нашими сторінками в соцмережах:</p>
            </div>
            <div className="d-flex flex-column justify-content-center">
                <p><i className='fab fa-youtube'></i> <a
href="https://www.youtube.com/channel/UCya7l0Wl8SqlLCl5Jh3J04A">Youtube-канал</a></p>
                <p><i className='fab fa-telegram'></i> <a
href="https://t.me/ndrtprv">Telegram-канал</a></p>
```

```
                <p><i className='fa-solid fa-phone'></i> <a href="tel:+380668202729">+38 (066)
820-27-29</a></p>
            </div>
        </Container>
    );
}

export default OtherContacts;
```

Feedback.js

```
import axios from "axios";
import { useState } from "react";
import { Container, Form } from "react-bootstrap";

function Feedback(props) {

    const [feedbackData, setFeedbackData] = useState({
        name: "",
        email: "",
        topic: "",
        text: "",
    });

    const { name, email, topic, text } = feedbackData;

    const onChange = (e) => {
        setFeedbackData({ ...feedbackData, [e.target.name]: e.target.value });
    };

    const classNameContainer = !props.isSmall ? props.classNameContainer : "";

    const handleSendMessage = async (e) => {
        e.preventDefault();
        try {
            await axios.post(process.env.REACT_APP_API_URL + 'feedback/send', feedbackData)
            .then(response => {
                console.log(response);
                if (response.data.status) {
                    alert(response.data.message);
                }
            }).catch(err => {
                alert(err.message);
            });
        } catch (e) {
            alert(e.response.data.message);
        }
```

```
    }

    return (
        <Container className={classNameContainer}>
            {
                !props.isSmall ?
                <>
                    <h2 className="featurette-heading lat-h2">{props.headerFeedback}</h2>
                    <p className="lead">{props.textFeedback}</p>
                </>
                :
                <></>
            }
            <Form method="post" id="form-box" className={props.classNameForm}
onSubmit={handleSendMessage}>
                <Form.Group className="input-group mb-2">
                    <div className="input-group-prepend">
                        <span className="input-group-text"><i className="fas fa-user"></i>
<b><span style={{color: "red"}}>*</span></b></span>
                    </div>
                    <Form.Control type="text" name="name" placeholder="Введіть ваше ім'я"
value={name} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="input-group mt-2 mb-2">
                    <div className="input-group-prepend">
                        <span className="input-group-text"><i className="fas fa-envelope"></i>
<b><span style={{color: "red"}}>*</span></b></span>
                    </div>
                    <Form.Control type="email" name="email" placeholder="Введіть ваш email"
value={email} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="input-group mt-2 mb-2">
                    <div className="input-group-prepend">
                        <span className="input-group-text"><i className="fas fa-at"></i>
<b><span style={{color: "red"}}>*</span></b></span>
                    </div>
                    <Form.Control type="text" name="topic" placeholder="Тема листа"
value={topic} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="input-group mt-2 mb-2">
                    <div className="input-group-prepend">
                        <span className="input-group-text"><i className="fas fa-comment-
alt"></i> <b><span style={{color: "red"}}>*</span></b></span>
```

```
                    </div>
                    <Form.Control as="textarea" type="text" name="text" id="text"
placeholder="Ваше повідомлення..." cols="30" rows="4" value={text} onChange={onChange}
required></Form.Control>
                </Form.Group>

                <Form.Group className="mt-2">
                    <Form.Control type="submit" name="submit" id="submit" className="btn btn-
primary btn-block"
                        disabled={name === "" || email === "" || topic === "" || text === ""}
value="Відправити"
                    />
                </Form.Group>

                <Form.Group className="mt-3 registration-field" controlId="formBasicTip">
                    <Form.Label><b><span style={{color: "red"}}>*</span></b> - обов'язково до
заповнення</Form.Label>
                </Form.Group>
            </Form>
        </Container>
    );
}

export default Feedback;
```

FAQ.js

```
import Accordion from 'react-bootstrap/Accordion';
import './FAQ.css';
import { Container } from 'react-bootstrap';

function FAQ(props) {
    return(
        <main>
            <Container className="mt-3">
                <h2>Часто задавані питання</h2>
            </Container>
            <Accordion>
                {props.quests.map((quest, index) =>
                    <Accordion.Item key={index} eventKey={'' + index}>
                        <Accordion.Header>{quest.question}</Accordion.Header>
                        <Accordion.Body>
                            {quest.answer}
                        </Accordion.Body>
                    </Accordion.Item>
                )}
```

```
            </Accordion>
        </main>
    );
}

export default FAQ;
```

Registration.js

```
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import './Registration.css';
import React, { useEffect, useState } from 'react';
import {useNavigate} from 'react-router-dom';
import { Container } from 'react-bootstrap';
import { NavLink } from 'react-router-dom';
import { LOGIN_ROUTE, USER_ROUTE } from '../../utils/constants';
import axios from 'axios';

import { LANDING_ROUTE } from '../../utils/constants';

function Registration() {

    const navigate = useNavigate();

    const [formData, setFormData] = useState({
        login: "",
        phone_num: "",
        password: "",
        name: "",
        surname: "",
        bio: "",
        avatar: undefined,
        isAdminCandidate: "",
        hideData: ""
    });

    const { login, phone_num, password, name, surname, bio, avatar, isAdminCandidate, hideData
} = formData;

    const onChange = (e) => {
        if (e.target.name === "avatar") {
            setFormData({ ...formData, [e.target.name]: e.target.files[0] })
        } else {
            setFormData({ ...formData, [e.target.name]: e.target.value })
            if (e.target.name === "password") {
```

```
                if (!e.target.value.match(/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[^A-Za-z\d])[A-
Za-z\d\W]{8,255}$/)) {
                    setDisableButton(true);
                } else {
                    setDisableButton(false);
                }
            }
        }
    };

    const [disableButton, setDisableButton] = useState(true);

    const [confirmationPassword, setConfirmationPassword] = useState("");

    const [agreement, setAgreement] = useState(false);

    axios.defaults.withCredentials = true;
    useEffect(() => {
        axios.get(process.env.REACT_APP_API_URL + 'user/nav')
        .then(res => {
            if (res.data.status) {
                navigate(USER_ROUTE);
            }
        }).catch(err => {
            console.log(err.message);
        })
    });

    const signup = async (e) => {
        e.preventDefault();
        try {
            const form = new FormData();
            form.append('login', login);
            form.append('phone_num', phone_num);
            form.append('password', password);
            form.append('name', name);
            form.append('surname', surname);
            form.append('bio', bio);
            form.append('isAdminCandidate', isAdminCandidate);
            form.append('hideData', hideData);

            if (avatar) {
                form.append('avatar', avatar);
            }

            await axios.post(process.env.REACT_APP_API_URL + 'user/signup',
```

```
                form,
                {
                    headers: {
                        "Content-Type": "multipart/form-data"
                    }
                }
            ).then(response => {
                if (response.data.status) {
                    navigate(LANDING_ROUTE);
                    alert("Повідомлення про підтвердження даних надіслано на вашу пошту.
Дійсне 15 хвилин.");
                }
            }).catch(err => {
                alert(err.message);
            });
        } catch (e) {
            alert(e.response.data.message);
        }
    }

    return (
        <main style={{display: 'inline-flex', flexDirection: 'column'}} >

            <Form method="post">
                <h2>Реєстрація</h2>
                <Form.Group className="mb-3 registration-field" controlId="formBasicEmail">
                    <Form.Label><b>Email <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="email" placeholder="Введіть Ваш email" name="login"
value={login} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicNumber">
                    <Form.Label><b>Номер телефону <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="tel" placeholder="Введіть Ваш номер телефону"
name="phone_num" value={phone_num} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicName">
                    <Form.Label><b>Ім'я <span style={{color: "red"}}>*</span></b></Form.Label>
                    <Form.Control type="text" placeholder="Введіть Ваше ім'я" name="name"
value={name} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicSurname">
```

```
                    <Form.Label><b>Прізвище <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="text" placeholder="Введіть Ваше прізвище"
name="surname" value={surname} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicBio">
                    <Form.Label><b>Додатково</b></Form.Label>
                    <Form.Control as="textarea" type="text" resize="none" placeholder="Введіть
додаткові дані про вас" name="bio" value={bio} onChange={onChange} />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicAvatar">
                    <Form.Label><b>Фото профілю</b></Form.Label>
                    <Form.Control type="file" accept="image/*" alt="Оберіть фото"
name="avatar" onChange={onChange} />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicPassword">
                    <Form.Label><b>Пароль <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="password" placeholder="Введіть пароль" name="password"
                        pattern="/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[^A-Za-z\d])[A-Za-
z\d\W]{8,255}$/"
                        value={password} onChange={onChange} required
                    />
                </Form.Group>

                <Form.Group className="mb-3 registration-field"
controlId="formBasicConfirmPassword">
                    <Form.Label><b>Підтвердження пароля <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="password" placeholder="Введіть пароль ще раз"
name="confirmation_password"
                        value={confirmationPassword} onChange={(e) =>
setConfirmationPassword(e.target.value)}  required
                    />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicRole">
                    <Form.Label><b>Роль <span style={{color: "red"}}>*</span></b></Form.Label>
                    <Form.Check type="radio" name={`isAdminCandidate`} label="Користувач"
value={false} onClick={onChange} required />
                    <Form.Check type="radio" name={`isAdminCandidate`} label="Адмін"
value={true} onClick={onChange} />
                </Form.Group>
```

```
                <Form.Group className="mb-3 registration-field" controlId="formBasicHide">
                    <Form.Label><b>Приховати дані? <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Check type="radio" name={`hideData`} label="Так" value={true}
onClick={onChange} required />
                    <Form.Check type="radio" name={`hideData`} label="Ні" value={false}
onClick={onChange} />
                </Form.Group>

                <Form.Group className="mb-3 registration-field" controlId="formBasicCheckbox">
                    <Form.Check type="checkbox" label="Я приймаю умови використання даного
сайту" name="agreement"
                        checked={agreement} onChange={(e) => setAgreement(e.target.checked)}
required
                    />
                </Form.Group>

                <Button variant="primary" type="submit"
                    disabled={
                        login === "" || phone_num === "" ||
                        name === "" || surname === "" ||
                        isAdminCandidate === "" || hideData === "" ||
                        disableButton || password !== confirmationPassword ||
                        password === "" || confirmationPassword === "" ||
                        !agreement
                    }
                    onClick={signup}
                >
                    Зареєструватися
                </Button>

                <Form.Group className="mt-3 registration-field" controlId="formBasicTip">
                    <Form.Label><b><span style={{color: "red"}}>*</span></b> - обов'язково до
заповнення</Form.Label>
                    <Form.Label>Пароль має містити не менше 8 символів</Form.Label>
                    <Form.Label>Пароль має містити мінімум 1 велику літеру</Form.Label>
                    <Form.Label>Пароль має містити мінімум 1 малу літеру</Form.Label>
                    <Form.Label>Пароль має містити мінімум 1 цифру</Form.Label>
                    <Form.Label>Пароль має містити мінімум 1 символ</Form.Label>
                </Form.Group>
            </Form>

            <Container style={{backgroundColor: 'beige', borderRadius: '0.5em', display:
'inline-flex', flexDirection: 'column'}} >
                <p>Уже зареєстровані? <NavLink to={LOGIN_ROUTE} >Авторизуватися</NavLink></p>
```

```
            </Container>
        </main>
    );
}

export default Registration;
```

Confirm.js

```
import React, { useEffect } from 'react'
import { useNavigate, useParams } from 'react-router-dom';
import axios from 'axios';
import { USER_ROUTE } from '../../utils/constants';

function Confirm() {

    const {token} = useParams();
    const navigate = useNavigate();

    useEffect(() => {
        axios.post(process.env.REACT_APP_API_URL + 'user/confirm/' + token)
        .then(response => {
            if (response.data.status) {
                alert(response.data.message);
                navigate(USER_ROUTE);
            }
        })
        .catch(err => {
            console.log(err.message);
        });
    });

    return (
        <div>Confirm</div>
    )
}

export default Confirm
```

Login.js

```
import { useEffect, useState } from 'react';
import {useNavigate} from 'react-router-dom';
import { Button, Form } from 'react-bootstrap';
import axios from 'axios';
import './Login.css';
```

```
import { Container} from 'react-bootstrap';
import { NavLink } from 'react-router-dom';
import { SIGNUP_ROUTE, FORGOT_ROUTE, USER_ROUTE } from '../../utils/constants';

function Login() {

    const navigate = useNavigate();

    const [formData, setFormData] = useState({
        login: "",
        password: ""
    });

    const { login, password } = formData;

    const onChange = (e) =>
        setFormData({ ...formData, [e.target.name]: e.target.value });

    axios.defaults.withCredentials = true;
    useEffect(() => {
        axios.get(process.env.REACT_APP_API_URL + 'user/nav')
        .then(res => {
            if (res.data.status) {
                navigate(USER_ROUTE);
            }
        }).catch(err => {
            alert(err.message);
        })
    });

    const logIn = async (e) => {
        e.preventDefault();
        try {
            await axios.post(process.env.REACT_APP_API_URL + 'user/login',
                {
                    login, password
                }
            ).then(response => {
                if (response.data.status) {
                    navigate(USER_ROUTE);
                }
            }).catch(err => {
                alert(err.message);
            });
        } catch (e) {
            alert(e.response.data.message);
```

```
        }
    }

    return (
        <main style={{display: 'inline-flex', flexDirection: 'column'}} >

            <Form method="post">
                <h2>Вхід до системи</h2>
                <Form.Group className="mb-3 login-field" controlId="formBasicEmail">
                    <Form.Label><b>Email <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="email" placeholder="Введіть email" name="login"
value={login} onChange={onChange} required />
                </Form.Group>

                <Form.Group className="mb-3 login-field" controlId="formBasicPassword">
                    <Form.Label><b>Пароль <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="password" placeholder="Введіть пароль" name="password"
value={password} onChange={onChange} required />
                </Form.Group>

                <Button variant="primary" type="submit" disabled={login === "" || password ===
""} onClick={logIn}>
                    Увійти
                </Button>

                <Form.Group className="mt-3 registration-field" controlId="formBasicTip">
                    <Form.Label><b><span style={{color: "red"}}>*</span></b> - обов'язково до
заповнення</Form.Label>
                </Form.Group>
            </Form>

            <Container style={{backgroundColor: 'beige', borderRadius: '0.5em', display:
'inline-flex', flexDirection: 'column'}} >
                <p><NavLink to={FORGOT_ROUTE} >Забули пароль?</NavLink></p>
                <p>Не маєте користувача? <NavLink to={SIGNUP_ROUTE}
>Зареєструватися</NavLink></p>
            </Container>
        </main>
    );
}

export default Login;
```

ForgotPassword.js

```
import React, { useEffect, useState } from 'react';
import { Button, Container, Form } from 'react-bootstrap';
import axios from 'axios';
import { USER_ROUTE } from '../../utils/constants';
import { useNavigate } from 'react-router-dom';

function ForgotPassword() {

  const navigate = useNavigate();

  const [login, setLogin] = useState("");
  const [notification, setNotification] = useState(undefined);

  axios.defaults.withCredentials = true;
  useEffect(() => {
    axios.get(process.env.REACT_APP_API_URL + 'user/nav')
    .then(res => {
      if (res.data.status) {
        navigate(USER_ROUTE);
      }
    }).catch(err => {
      console.log(err.message);
    })
  });

  const forgotPassword = async (e) => {
    try {
      e.preventDefault();
      await axios.post(process.env.REACT_APP_API_URL + 'user/forgot-password',
        {
          login
        }
      ).then(response => {
        if (response.data.status) {
          setNotification('Перевірте пошту, на яке надійшло посилання для скидання паролю.');
        }
      })
      .catch(err => {
        console.log(err.message);
      });
    } catch (e) {
      alert(e.response.data.message);
    }
  }
```

```
  return (
    <main className='mt-5'>
      {notification !== undefined ?
        <Container style={{backgroundColor: 'green', color: 'white'}}>
          <b>{notification}</b>
        </Container>
        : <></>
      }

      <Form method="post" >
        <h2>Забули пароль?</h2>

        <Form.Group className="mb-3 login-field" controlId="formBasicEmail">
          <Form.Label><b>Ваш email</b></Form.Label>
          <Form.Control type="email" placeholder="Введіть ваш email" name="login"
value={login} onChange={(e) => setLogin(e.target.value)} required />
        </Form.Group>

        <Button variant="primary" type="submit" onClick={forgotPassword} disabled={login ===
""}>
          Відправити посилання на пошту
        </Button>
      </Form>
    </main>
  );
}

export default ForgotPassword;
```

ResetPassword.js

```
import React, { useEffect, useState } from 'react';
import { Button, Form } from 'react-bootstrap';
import {useNavigate, useParams} from 'react-router-dom';
import axios from 'axios';
import { LOGIN_ROUTE, USER_ROUTE } from '../../utils/constants';

function ResetPassword() {

    const navigate = useNavigate();
    const {token} = useParams();

    const [password, setPassword] = useState("");
    const [confirmationPassword, setConfirmationPassword] = useState("");
```

```
    axios.defaults.withCredentials = true;
    useEffect(() => {
        axios.get(process.env.REACT_APP_API_URL + 'user/nav')
        .then(res => {
            if (res.data.status) {
                navigate(USER_ROUTE);
            }
        }).catch(err => {
            console.log(err.message);
        })
    });

    const resetPassword = async (e) => {
        try {
            e.preventDefault();
            await axios.post(process.env.REACT_APP_API_URL + 'user/reset-password/' + token,
                {
                    password
                }
            ).then(response => {
                if (response.data.status) {
                    navigate(LOGIN_ROUTE);
                }
            })
            .catch(err => {
                console.log(err.message);
            });
        } catch (e) {
            alert(e.response.data.message);
        }
    }

    return (
        <main className='mt-5'>

            <Form method="post">
                <h2>Змінити пароль</h2>

                <Form.Group className="mb-3 registration-field" controlId="formBasicPassword">
                    <Form.Label><b>Новий пароль <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="password" placeholder="Введіть пароль" name="password"
value={password} onChange={(e) => setPassword(e.target.value)} required />
                </Form.Group>

                <Form.Group className="mb-3 registration-field"
```

```
controlId="formBasicConfirmPassword">
                    <Form.Label><b>Підтвердження нового пароля <span style={{color:
"red"}}>*</span></b></Form.Label>
                    <Form.Control type="password" placeholder="Введіть пароль ще раз"
name="confirmation_password"
                        value={confirmationPassword} onChange={(e) =>
setConfirmationPassword(e.target.value)}  required
                    />
                </Form.Group>

                <Button variant="primary" type="submit"
                    disabled={password !== confirmationPassword || password === "" ||
confirmationPassword === ""}
                    onClick={resetPassword}
                >
                    Змінити пароль
                </Button>

                <Form.Group className="mt-3 registration-field" controlId="formBasicTip">
                    <Form.Label><b><span style={{color: "red"}}>*</span></b> - обов'язково до
заповнення</Form.Label>
                </Form.Group>
            </Form>
        </main>
    );
}

export default ResetPassword
```

UserPanel.js

```
import axios from 'axios';
import React, { useEffect, useState } from 'react';
import { Col, Container, Row, Button, Form, Accordion } from 'react-bootstrap';
import Avatar from 'react-avatar';
import { useNavigate } from 'react-router-dom';
import { LANDING_ROUTE } from '../../utils/constants';
import './UserPanel.css';
import ChangeForm from '../../components/change_form/ChangeForm';
import NoticeForm from '../../components/notice_form/NoticeForm';
import DeleteUser from '../../components/modals/DeleteUser';
//import NoticeList from '../../components/notice_list/NoticeList';

function UserPanel(props) {

  const navigate = useNavigate();
```

```
const [verifiedData, setVerifiedData] = useState(true);
const [avatar, setAvatar] = useState(props.avatar);
const [userData, setUserData] = useState({});

axios.defaults.withCredentials = true;
useEffect(() => {
  axios.get(process.env.REACT_APP_API_URL + 'user/profile')
  .then(res => {
    if (res.data.status) {
      setUserData(res.data.user);

      if (res.data.user.verifiedAt === null) {
        setVerifiedData(false);
      }

      if (res.data.portrait !== null) {
        setAvatar(res.data.portrait);
      }
    } else {
      navigate(LANDING_ROUTE);
    }
  })
});

const handleSend = async (e) => {
  e.preventDefault();
  try {
    await axios.post(process.env.REACT_APP_API_URL + 'user/resend')
    .then(response => {
      console.log(response);
      if (response.data.status) {
        alert(response.data.message);
      }
    }).catch(err => {
      alert('Не вдалося надіслати листа за вказаною адресою. ' + err.message);
    });
  } catch (e) {
    alert(e.response.data.message);
  }
}

return (
  <main>
    <Container className="mt-5">
      <Row>
```

```
          <Col className="d-flex flex-column align-items-center justify-content-center" md={4}
>
            <Avatar alt="Профіль" variant='circular' src={avatar !== props.avatar ?
`data:${avatar.contentType};base64,${avatar.data}` : props.avatar} size="15em" />
          </Col>
          <Col md >
            <Container className="justify-content-center">
              <h3>{userData.name} {userData.surname}</h3>
            </Container>
            <Container className={verifiedData ? "justify-content-center" : "justify-content-
center m-5"}>
              {
                verifiedData ?
                <ChangeForm userData={userData} />
                :
                <Form action="post" className="m-12">
                  <h3><b>Ви не підтвердили свої дані. Зробіть це зараз.</b></h3>
                  <Button variant="primary" type="submit" onClick={handleSend} >
                    Підтвердити дані
                  </Button>
                </Form>
              }
              <DeleteUser />
            </Container>
          </Col>
        </Row>
        {
          verifiedData ?
          <>
          <Accordion>
            <Accordion.Item eventKey="0">
              <Accordion.Header>Форма додавання оголошення</Accordion.Header>
              <Accordion.Body>
                <NoticeForm />
              </Accordion.Body>
            </Accordion.Item>

            <Accordion.Item eventKey="1">
              <Accordion.Header>Форма додавання збору</Accordion.Header>
              <Accordion.Body>
                <Form action="post">

                </Form>
              </Accordion.Body>
            </Accordion.Item>
```

```
            <Accordion.Item eventKey="2">
              <Accordion.Header>Форма додавання звіту</Accordion.Header>
              <Accordion.Body>
                <Form action="post">

                </Form>
              </Accordion.Body>
            </Accordion.Item>
          </Accordion>

          </>
          :
          <></>
        }
      </Container>
    </main>
  )
}

export default UserPanel;
```

DeleteUser.js

```
import axios from 'axios';
import React, { useState } from 'react';
import { Button, Modal } from 'react-bootstrap';

function DeleteUser() {

    const [show, setShow] = useState(false);

    const handleClose = () => setShow(false);
    const handleShow = () => setShow(true);

    const handleDelete = async (e) => {
        e.preventDefault();
        try {
            await axios.delete(process.env.REACT_APP_API_URL + 'user/deleteUser')
            .then(response => {
                alert(response.data.message);
            }).catch(err => {
                alert(err.message);
            });
        } catch (e) {
          alert(e.response.data.message);
        }
```

```
    }

    return (
        <>
        <Button className='btn-warning' onClick={handleShow}>Видалити користувача</Button>
        <Modal show={show} onHide={handleClose}>
            <Modal.Header closeButton>
                <Modal.Title>Видалення користувача</Modal.Title>
            </Modal.Header>
            <Modal.Body>Ви дійсно хочете видалити користувача?</Modal.Body>
            <Modal.Footer>
                <Button variant="secondary" onClick={handleClose}>
                    Відмінити
                </Button>
                <Button className='btn-warning' variant="primary" onClick={handleDelete}>
                    Видалити
                </Button>
            </Modal.Footer>
        </Modal>
        </>
    )
}

export default DeleteUser
```

ДОДАТОК Б

Серверна частина сайту вебзастосунку

index.js

```
require('dotenv').config();
const express = require('express');
const sequelize = require('./db');
const models = require('./models/models');
const cors = require('cors');
const cookieParser = require('cookie-parser');
const router = require('./routes/index');
const errorHandler = require('./middleware/ErrorHandlingMiddleware');

const PORT = process.env.PORT;

const app = express();
app.use(cors({
    origin: [process.env.ORIGIN],
    methods: ["GET", "POST", "DELETE"],
    credentials: true
}));
app.options('*', cors());
app.use(express.json());
app.use(cookieParser());
app.use(router);

// Для обробки помилок
app.use(errorHandler);

const start = async () => {
    try {
        await sequelize.authenticate();
        await sequelize.sync();
        app.listen(PORT, () => console.log(`It's working on port ${PORT}!`));
    } catch (err) {
        console.log(err.message);
    }
}

start();
```

db.js

```
const { Sequelize } = require('sequelize');
```

```
module.exports = new Sequelize(
    process.env.DB_NAME,
    process.env.DB_USER,
    process.env.DB_PASSWORD,
    {
        dialect: "postgres",
        define: {timestamps: false},
        host: process.env.DB_HOST,
        port: process.env.DB_PORT
    }
);
```

models.js

```
const sequelize = require('../db');
const { DataTypes } = require('sequelize');

const User = sequelize.define('user', {
    id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    login: {type: DataTypes.STRING(320), allowNull: false, unique: true},
    phone_num: {type: DataTypes.STRING(50), allowNull: false},
    password: {type: DataTypes.STRING, allowNull: false},
    name: {type: DataTypes.STRING(300), allowNull: false},
    surname: {type: DataTypes.STRING(300), allowNull: false},
    bio: {type: DataTypes.TEXT},
    hideData: {type: DataTypes.BOOLEAN, allowNull: false},
    createdAt: {type: DataTypes.DATE, allowNull: false},
    verifiedAt: {type: DataTypes.DATE}
});

const Admin = sequelize.define('admin', {
    admin_id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    status: {type: DataTypes.SMALLINT, allowNull: false},
    role: {type: DataTypes.STRING, allowNull: false, defaultValue: "Адміністратор"}
});

const Avatars = sequelize.define('avatars', {
    id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    data: {type: DataTypes.BLOB, allowNull: false},
    contentType: {type: DataTypes.STRING, allowNull: false}
});

const Notice = sequelize.define('notice', {
    id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    type: {type: DataTypes.SMALLINT, allowNull: false},
    kind: {type: DataTypes.STRING, allowNull: false},
```

```
    description: {type: DataTypes.TEXT, allowNull: false},
    createdAt: {type: DataTypes.DATE, allowNull: false}
});

const Fundraise = sequelize.define('fundraise', {
    id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    type: {type: DataTypes.SMALLINT, allowNull: false},
    whom: {type: DataTypes.STRING, allowNull: false},
    kind: {type: DataTypes.STRING, allowNull: false},
    description: {type: DataTypes.TEXT, allowNull: false},
    target_sum: {type: DataTypes.DECIMAL(2), allowNull: false},
    status: {type: DataTypes.SMALLINT, allowNull: false},
    createdAt: {type: DataTypes.DATE, allowNull: false}
});

const Result = sequelize.define('result', {
    id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    src_photo: {type: DataTypes.BLOB, allowNull: false},
    description: {type: DataTypes.STRING, allowNull: false},
    createdAt: {type: DataTypes.DATE, allowNull: false}
});

const Photo = sequelize.define('photo', {
    photo_id: {type: DataTypes.BIGINT, primaryKey: true, autoIncrement: true},
    src_photo: {type: DataTypes.BLOB, allowNull: false},
    contentType: {type: DataTypes.STRING, allowNull: false}
});

User.hasOne(Admin, {foreignKey: 'user_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Admin.belongsTo(User, {foreignKey: 'user_id', targetKey: 'id', onDelete: 'cascade', onUpdate:
'cascade'});

User.hasOne(Avatars, {foreignKey: 'user_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Avatars.belongsTo(User, {foreignKey: 'user_id', targetKey: 'id', onDelete: 'cascade',
onUpdate: 'cascade'});

User.hasMany(Notice, {foreignKey: 'user_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Notice.belongsTo(User, {foreignKey: 'user_id', targetKey: 'id', onDelete: 'cascade', onUpdate:
'cascade'});

User.hasMany(Fundraise, {foreignKey: 'user_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Fundraise.belongsTo(User, {foreignKey: 'user_id', targetKey: 'id', onDelete: 'cascade',
```

```
onUpdate: 'cascade'});

Notice.hasMany(Photo, {foreignKey: 'notice_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Photo.belongsTo(Notice, {foreignKey: 'notice_id', targetKey: 'id', onDelete: 'cascade',
onUpdate: 'cascade'});

Fundraise.hasMany(Photo, {foreignKey: 'fundraise_id', sourceKey: 'id', onDelete: 'cascade',
hooks: true});
Photo.belongsTo(Fundraise, {foreignKey: 'fundraise_id', targetKey: 'id', onDelete: 'cascade',
onUpdate: 'cascade'});

Fundraise.hasOne(Result, {foreignKey: 'fundraise_id', sourceKey: 'id', onDelete: 'cascade',
hooks: true});
Result.belongsTo(Fundraise, {foreignKey: 'fundraise_id', targetKey: 'id', onDelete: 'cascade',
onUpdate: 'cascade'});

Result.hasMany(Photo, {foreignKey: 'result_id', sourceKey: 'id', onDelete: 'cascade', hooks:
true});
Photo.belongsTo(Result, {foreignKey: 'result_id', targetKey: 'id', onDelete: 'cascade',
onUpdate: 'cascade'});

module.exports = {
    User,
    Admin,
    Avatars,
    Notice,
    Fundraise,
    Result,
    Photo
};
```

routes/index.js

```
const Router = require('express');
const router = new Router();
const userRouter = require('./userRouter');
const feedbackRouter = require('./feedbackRouter')
const noticeRouter = require('./noticeRouter');

router.use('/user', userRouter);
router.use('/feedback', feedbackRouter);
router.use('/notice', noticeRouter);

module.exports = router;
```

userRouter.js

```
const Router = require('express');
const multer = require('multer');
const router = new Router();
const userController = require('../controllers/UserController');
const { body } = require('express-validator');
const jwt = require('jsonwebtoken');

const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

const generateAccessJwt = (login, phone_num, name, surname) => {
    return jwt.sign(
        {login: login, phone_num: phone_num, name: name, surname: surname},
        process.env.SECRET_ACCESS_KEY,
        {expiresIn: '15m'}
    );
}

const verifyUser = async (req, res, next) => {
    const accessToken = req.cookies.accessToken;
    const refreshToken = req.cookies.refreshToken;
    if (!accessToken) {
        if (!refreshToken) {
            return res.json({status: false, message: 'Ви не авторизовані!'});
        } else {
            await jwt.verify(refreshToken, process.env.SECRET_REFRESH_KEY, (err, decoded) => {
                if (err) {
                    return res.json({status: false, message: 'Невалідний токен оновлення!'});
                } else {
                    const accessToken = generateAccessJwt(decoded.login, decoded.phone_num,
decoded.name, decoded.surname);
                    res.cookie('accessToken', accessToken, {maxAge: 900000});
                    req.login = decoded.login;
                    next();
                }
            });
        }
    } else {
        await jwt.verify(accessToken, process.env.SECRET_ACCESS_KEY, (err, decoded) => {
            if (err) {
                return res.json({status: false, message: 'Невалідний токен доступу!'});
            } else {
                req.login = decoded.login;
                next();
```

```
            }
        });
    }
}

router.post('/signup', body('login').isEmail(), upload.single('avatar'),
userController.registration);
router.post('/resend', userController.resend);
router.post('/login', body('login').isEmail(), userController.login);
router.post('/forgot-password', body('login').isEmail(), userController.forgotPassword);
router.post('/reset-password/:token', userController.resetPassword);
router.post('/confirm/:token', userController.confirm);
router.post('/updateData', verifyUser, userController.updateData)
router.get('/nav', verifyUser, userController.getNavigation)
router.get('/profile', verifyUser, userController.getProfile);
router.get('/logout', userController.logout);
router.delete('/deleteUser', verifyUser, userController.delete)

module.exports = router;
```

UserController.js

```
const ApiError = require('../error/ApiError');
const bcrypt = require('bcrypt');
const { User, Admin, Avatars } = require('../models/models');
const jwt = require('jsonwebtoken');
const { validationResult } = require('express-validator');
const nodemailer = require('nodemailer');

const generateAccessJwt = (login, phone_num, name, surname) => {
    return jwt.sign(
        {login: login, phone_num: phone_num, name: name, surname: surname},
        process.env.SECRET_ACCESS_KEY,
        {expiresIn: '15m'}
    );
}

const generateRefreshJwt = (login, phone_num, name, surname) => {
    return jwt.sign(
        {login: login, phone_num: phone_num, name: name, surname: surname},
        process.env.SECRET_REFRESH_KEY,
        {expiresIn: '8h'}
    );
}

const generateJwtForMail = (login, phone_num, name, surname) => {
```

```
    return jwt.sign(
        {login: login, phone_num: phone_num, name: name, surname: surname},
        process.env.SECRET_ACCESS_KEY,
        {expiresIn: '15m'}
    );
}

class UserController {
    async registration(req, res, next) {

        const { login, phone_num, password, name, surname, bio, isAdminCandidate, hideData } =
req.body;

        try {
            const createdAt = new Date();
            const errors = validationResult(req.body);

            if (!errors.isEmpty()) {
                return next(ApiError.badRequest("Невалідний email!"));
            }

            if (!login || !phone_num || !password || !name || !surname) {
                return next(ApiError.badRequest("Не введено дані!"));
            }

            const candidate = await User.findOne({where: {login}});
            if (candidate) {
                return next(ApiError.badRequest("Користувач уже існує в системі!"));
            }

            const encryptedPassword = await bcrypt.hash(password, 7);

            const newUser = await User.create({
                login,
                phone_num,
                password: encryptedPassword,
                name,
                surname,
                bio,
                hideData,
                createdAt,
                verifiedAt: null
            });

            if (req.file !== undefined) {
                await Avatars.create({
```

```
                data: req.file.buffer,
                contentType: req.file.mimetype,
                user_id: newUser.id
            });
        }

        if (isAdminCandidate) {
            await Admin.create({
                status: 0,
                user_id: newUser.id
            });
        }

        const accessToken = generateAccessJwt(newUser.login, newUser.phone_num,
newUser.name, newUser.surname);
        const refreshToken = generateRefreshJwt(newUser.login, newUser.phone_num,
newUser.name, newUser.surname);

        res.cookie('accessToken', accessToken, {maxAge: 900000});
        res.cookie('refreshToken', refreshToken, {maxAge: 3600000 * 8, httpOnly: true,
secure: true, sameSite: 'strict'});

        const mailToken = generateJwtForMail(newUser.login, newUser.phone_num,
newUser.name, newUser.surname);

        var transporter = nodemailer.createTransport({
            service: 'gmail',
            auth: {
                user: process.env.ORIGIN_EMAIL,
                pass: process.env.ORIGIN_PASSWORD
            }
        });

        const encodedToken = encodeURIComponent(mailToken).replace(/\./g, "%2E");
        const link = `http://localhost:3000/confirm/${encodedToken}`;

        var mailOptions = {
            from: process.env.ORIGIN_EMAIL,
            to: login,
            subject: 'Підтвердження даних',
            html: `<div>
                        <p>Вам надіслано повідомлення на підтвердження даних. Якщо це були
не ви, повідомте нам: ми видалимо користувача.</p>
                        <a href=${link} class="btn btn-primary">Підтвердити дані</a>
                    </div>`
        };
```

```
            await transporter.sendMail(mailOptions, function(error, info){
                if (error) {
                    return res.json({message: 'Помилка надсилання листа за вказаним email!'});
                }
            });

            return res.json({status: true, message: "Реєстрація успішна!"});
        } catch(e) {
            next(ApiError.internal(e.message));
        }
    }

    async resend(req, res, next) {
        const accessToken = req.cookies.accessToken;

        try {
            const decoded = jwt.verify(accessToken, process.env.SECRET_ACCESS_KEY);
            const user = await User.findOne({where: {login: decoded.login}});

            const mailToken = generateJwtForMail(user.login, user.phone_num, user.name,
user.surname);

            var transporter = nodemailer.createTransport({
                service: 'gmail',
                auth: {
                    user: process.env.ORIGIN_EMAIL,
                    pass: process.env.ORIGIN_PASSWORD
                }
            });

            const encodedToken = encodeURIComponent(mailToken).replace(/\./g, "%2E");
            const link = `http://localhost:3000/confirm/${encodedToken}`;

            var mailOptions = {
                from: process.env.ORIGIN_EMAIL,
                to: user.login,
                subject: 'Підтвердження даних',
                html: `<div>
                            <p>Вам надіслано повідомлення на підтвердження даних. Якщо це були
не ви, повідомте нам: ми видалимо користувача.</p>
                            <a href=${link} class="btn btn-primary">Підтвердити дані</a>
                        </div>`
            };

            await transporter.sendMail(mailOptions, function(error, info){
```

```
                if (error) {
                    return res.json({message: 'Помилка надсилання листа за вказаним email!'});
                } else {
                    return res.json({status: true, message: 'Лист надіслано за вказаним email:
' + info.response});
                }
            });
        } catch(e) {
            next(ApiError.internal(e.message + req.cookies.accessToken));
        }
    }

    async confirm(req, res, next) {
        const {token} = req.params;
        console.log(token);

        try {
            const decoded = jwt.verify(token, process.env.SECRET_ACCESS_KEY);
            const verifiedAt = new Date();

            await User.update({verifiedAt: verifiedAt}, {where: {login: decoded.login}});
            return res.json({status: true, message: 'Користувача підтверджено! Вітаю!'});
        } catch(e) {
            next(ApiError.internal(e.message));
        }
    }

    async login(req, res, next) {
        const { login, password } = req.body;

        try {
            const errors = validationResult(req);

            if (!errors.isEmpty()) {
                return next(ApiError.badRequest("Невалідний email!"));
            }

            const user = await User.findOne({where: {login}});
            if (!user) {
                return next(ApiError.badRequest("Користувач не зареєстрований!"));
            }

            let checkPassword = bcrypt.compareSync(password, user.password);
            if (!checkPassword) {
                return next(ApiError.badRequest("Пароль неправильний!"));
            }
```

```
            const accessToken = generateAccessJwt(user.login, user.phone_num, user.name,
user.surname);
            const refreshToken = generateRefreshJwt(user.login, user.phone_num, user.name,
user.surname);

            res.cookie('accessToken', accessToken, {maxAge: 900000});
            res.cookie('refreshToken', refreshToken, {maxAge: 3600000 * 8, httpOnly: true,
secure: true, sameSite: 'strict'});

            return res.json({status: true, message: "Авторизація успішна!"});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async forgotPassword(req, res, next) {
        const {login} = req.body;

        try {
            const errors = validationResult(req);

            if (!errors.isEmpty()) {
                return next(ApiError.badRequest("Невалідний email!"));
            }

            const user = await User.findOne({where: {login}});
            if (!user) {
                return next(ApiError.badRequest("Користувач не зареєстрований!"));
            }

            const token = generateJwtForMail(user.login, user.phone_num, user.name,
user.surname);

            var transporter = nodemailer.createTransport({
                service: 'gmail',
                auth: {
                    user: process.env.ORIGIN_EMAIL,
                    pass: process.env.ORIGIN_PASSWORD
                }
            });

            const encodedToken = encodeURIComponent(token).replace(/\./g, "%2E");
            const link = `http://localhost:3000/resetPassword/${encodedToken}`;

            var mailOptions = {
```

```
                from: process.env.ORIGIN_EMAIL,
                to: login,
                subject: 'Відновлення паролю',
                html: `<div>
                            <p>Вам надіслано повідомлення на скидання паролю. Якщо це були не
ви, проігноруйте повідомлення.</p>
                            <a href=${link} class="btn btn-primary">Скинути пароль</a>
                        </div>`
            };

            await transporter.sendMail(mailOptions, function(error, info){
                if (error) {
                    return res.json({message: 'Помилка надсилання листа за вказаним email!'});
                } else {
                    return res.json({status: true, message: 'Лист надіслано за вказаним email:
' + info.response});
                }
            });
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async resetPassword(req, res, next) {
        const {token} = req.params;
        const {password} = req.body;

        try {
            const decoded = jwt.verify(token, process.env.SECRET_ACCESS_KEY);
            const encryptedPassword = await bcrypt.hash(password, 7);

            await User.update({password: encryptedPassword}, {where: {login: decoded.login}});

            return res.json({status: true, message: 'Пароль оновлено!'});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async updateData(req, res, next) {
        const { phone_num, name, surname, bio } = req.body;

        try {
            const user = await User.findOne({where: { login: req.login }});

            if (phone_num !== user.phone_num) {
```

```
                await User.update({phone_num: phone_num}, {where: {login: req.login}});
            }
            if (name !== user.name) {
                await User.update({name: name}, {where: {login: req.login}});
            }
            if (surname !== user.surname) {
                await User.update({surname: surname}, {where: {login: req.login}});
            }
            if (bio !== bio) {
                await User.update({bio: bio}, {where: {login: req.login}});
            }

            return res.json({status: true, message: "Оновлення даних успішно проведено!"});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async getNavigation(req, res, next) {
        try {

            const user = await User.findOne({where: {login: req.login}});
            const possibleAvatar = await Avatars.findOne({where: {user_id: user.id}});
            const possibleAdmin = await Admin.findOne({where: {user_id: user.id, status: 1}});

            let photo;
            if (possibleAvatar) {
                photo = {
                    data: possibleAvatar.data.toString("base64"),
                    contentType: possibleAvatar.contentType
                };
            } else {
                photo = null;
            }

            if (possibleAdmin) {
                return res.json({status: true, isAdmin: true, portrait: photo, message: 'Ви
авторизовані! Ви адмін.'});
            } else {
                return res.json({status: true, isAdmin: false, portrait: photo, message: 'Ви
авторизовані!'});
            }
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }
```

```
    async getProfile(req, res, next) {
        try {
            const user = await User.findOne({where: {login: req.login}});
            const possibleAvatar = await Avatars.findOne({where: {user_id: user.id}});

            let photo;
            if (possibleAvatar) {
                photo = {
                    data: possibleAvatar.data.toString("base64"),
                    contentType: possibleAvatar.contentType
                };
            } else {
                photo = null;
            }

            return res.json({status: true, user, portrait: photo, message: 'Ви
авторизовані!'});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async logout(req, res, next) {
        try {
            res.clearCookie('accessToken');
            res.clearCookie('refreshToken');
            return res.json({status: true, message: 'Вихід успішний!'});
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }

    async delete(req, res, next) {
        try {

            const user = await User.findOne({where: {login: req.login}});

            if (user) {
                await User.destroy({where: {login: user.login}});

                res.clearCookie('accessToken');
                res.clearCookie('refreshToken');

                return res.json({status: true, message: 'Видалення користувача успішне!'});
            } else {
```

```
                return res.json({status: false, message: 'Користувача не знайдено!'});
            }
        } catch (e) {
            next(ApiError.internal(e.message))
        }
    }
}

module.exports = new UserController();
```

noticeRouter.js

```
const Router = require('express');
const multer = require('multer');
const noticeController = require('../controllers/NoticeController');
const router = new Router();

const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

router.post('/add', upload.single('photo'), noticeController.addNotice);

module.exports = router;
```

NoticeController.js

```
const ApiError = require("../error/ApiError");
const jwt = require('jsonwebtoken');
const { Notice, Photo, User } = require("../models/models");

class NoticeController {

    async addNotice(req, res, next) {
        const { type, kind, description } = req.body;
        const file = req.file;
        const accessToken = req.cookies.accessToken;

        try {
            const createdAt = new Date();

            if (!type || !kind || !description) {
                return next(ApiError.badRequest("Не введено дані!"));
            }

            let chosenType = 0
            if (type === "Допомога ЗСУ") {
```

```
                chosenType = 0;
            } else {
                chosenType = 1;
            }

            const decoded = jwt.verify(accessToken, process.env.SECRET_ACCESS_KEY);
            const user = await User.findOne({where: {login: decoded.login}});

            const newNotice = await Notice.create({ type: chosenType, kind, description,
createdAt, user_id: user.id });

            await Photo.create({
                src_photo: file.buffer,
                contentType: file.mimetype,
                notice_id: newNotice.id,
                fundraise_id: null,
                result_id: null
            });

            return res.json({status: true, message: "Оголошення додане." })
        } catch (e) {
            next(ApiError.badRequest(e.message))
        }
    }
}

module.exports = new NoticeController();
```

feedbackRouter.js

```
const Router = require('express');
const nodemailer = require('nodemailer');
const ApiError = require('../error/ApiError');
const router = new Router();

router.post('/send', async (req, res, next) => {
    const { name, email, topic, text } = req.body;

    try {
        var transporter = nodemailer.createTransport({
            service: 'gmail',
            auth: {
                user: process.env.ORIGIN_EMAIL,
                pass: process.env.ORIGIN_PASSWORD
            }
        });
```

```
        var mailOptions = {
            from: email,
            to: process.env.ORIGIN_EMAIL,
            subject: name + ', ' + topic,
            text: text
        };

        await transporter.sendMail(mailOptions, function(error, info){
            if (error) {
                return res.json({message: 'Помилка надсилання листа за вказаним email!'});
            } else {
                return res.json({status: true, message: 'Лист надіслано за вказаним email.
Очікуйте на відповідь.'});
            }
        });
    } catch {
        next(ApiError.badRequest(e.message))
    }
})

module.exports = router;
```