

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ

(назва теми)

БЛАГОДІЙНОЇ ДОПОМОГИ

SOFTWARE FOR CHARITY SUPPORT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-219

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ЛАЗУТІН Д.Д.

(ПРІЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛАЗУТІНА Данііла Дмитровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для благодійної допомоги.
Software for Charity Support

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 12 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми.
3. Реалізація програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	БЕЛОВА А.В., асистент		

7. Дата видачі завдання « 11 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

_____ (підпис) _____ Данііл ЛАЗУТІН
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ (підпис) _____ Степан СКРУПСЬКИЙ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 97 с., 24 рис., 2 табл., 3 дод., 10 джерел.

БЛАГОДІЙНА ДОПОМОГА, БЛАГОДІЙНА ОРГАНІЗАЦІЯ, БЛАГОДІЙНИЙ ПРОЄКТ, БЛАГОДІЙНА ПОДІЯ, ВОЛОНТЕРСТВО, ВЕБДОДАТОК.

Об'єкт роботи – процес надання благодійної допомоги. Предмет роботи – програмне забезпечення для благодійної допомоги. Мета роботи – розробити програмне забезпечення для благодійної допомоги.

Матеріали, методи та технічні засоби: мова Python, фреймворк Django для веброзробки, система PostgreSQL для підтримки роботи з базами даних.

Результати. Розроблена програма призначена для підтримки роботи благодійників та благодійних організацій в частині отримання ресурсу для допомоги та знаходження потребуючих, щоби визначити, чи варто надавати їм допомогу, а також кому саме конкретну допомогу надавати.

Висновки. Використовуючи створене програмне забезпечення, благодійні організації можуть створювати благодійні проєкти, пропозиції проведення благодійних подій з продажем квитків на них за необхідності, визначити пропозиції для надання допомоги потребуючим, звичайні користувачі можуть виконувати пошук відповідних благодійних організацій, проєктів, подій, пропозицій, створювати власні пропозиції допомогти в їхній реалізації, подавати заявки на отримання допомоги, надавати власні пожертвування, просити про допомогу інших користувачів для вирішення конкретних проблем, а ті, в свою чергу, можуть надавати таку підтримку.

Галузь використання – благодійна допомога, волонтерство в частині підтримки благодійних організацій щодо збору коштів і пошуку людей для надання допомоги їм та їхніми силами.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 97 pages, 24 figures, 2 tables, 3 appendixes, 10 sources.

CHARITY AID, CHARITY ORGANIZATION, CHARITY PROJECT, CHARITY EVENT, VOLUNTEERING, WEB APPLICATION.

The object of the work is the process of providing charity support. The subject of the work is software for charity support. The purpose of the work is to develop software for charity support.

Materials, methods and technical means: Python language, Django framework for web development, PostgreSQL system to support work with databases.

The results. The developed program is intended to support the work of philanthropists and charitable organizations in terms of obtaining a resource for assistance and finding those in need, in order to determine whether it is worth providing assistance to them, as well as to whom specific assistance should be provided.

Conclusions. Using the created software, charitable organizations can create charitable projects, proposals for charity events with the sale of tickets for them if necessary, determine proposals for providing assistance to those in need, ordinary users can search for relevant charitable organizations, projects, events, proposals, create their own proposals to help in their implementation, submit applications for assistance, provide their own donations, ask for help from other users to solve specific problems, and they, in turn, can provide such support.

The field of use is charity support, volunteering in support of charitable organizations in terms of fundraising and finding people to help them and their forces.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Аналіз програмних засобів для благодійної допомоги	9
1.2 Визначення концепції розроблюваної програми	12
1.3 Висновки за розділом 1 та постановка завдань роботи	13
2 Розробка архітектури програми.....	14
2.1 Вибір технологій розробки	14
2.2 Моделювання вимог до програми	15
2.3 Проєктування бази даних	20
2.4 Висновки за розділом 2	35
3 Реалізація програми	36
3.1 Схема функціонування програми.....	36
3.2 Програмна реалізація функцій вебсистеми	37
3.3 Висновки за розділом 3	46
4 Експлуатація та тестування програми	47
4.1 Призначення програми	47
4.2 Умови виконання програми	47
4.3 Виконання і тестування програми.....	47
4.4 Висновки за розділом 4	53
Висновки	54
Перелік джерел посилання	55
Додаток А Технічне завдання	56
Додаток Б Текст програми	62
Додаток В Слайди презентації.....	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

URL – Uniform Resource Locator;

БД – база даних.

ВСТУП

З війною в Україні в певному стані, але спостерігається гуманітарна криза, а в деяких місцях, як наприклад, на території звільненої Херсонщини гуманітарний стан є катастрофічним. Тому на даний момент допомога благодійних організацій має критичне значення для виживання багатьох людей, які проживають в країні, а також і для тих, хто виїхав з країни, адже величезні кошти в світі витрачаються на підтримку біженців з України.

У 2020 році кошти, якими оперують благодійні організації, досягли позначки у майже 329,3 мільярдів доларів США, збільшившись при цьому на 6,9 % з 2015 року, а подальше збільшення прогнозується таким, що має в підсумку досягти 457,8 мільярдів доларів США у 2025 році [1]-[3]. Тобто це величезні кошти, а зі збільшенням коштів тільки ще більш критичною стає проблема розподілу їх, щоб відповідна гуманітарна допомога або будь-яка інша допомога, в яку вкладають кошти благодійні організації, потрапила в підсумку до тієї людини, якій вона дійсно потрібна.

Благодійна допомога має різні сторони і охоплює звичайно не тільки підтримку людей під час війни. Збір коштів є важливим для реалізації будь-яких проєктів, тому це має бути враховано в роботі. Але окрім цього благодійна організація може надавати допомогу тим, хто її потребує, визначаючи гуманітарну допомогу і шукаючи тих, у кого дійсно є потреба. Окрім того самі кандидати на допомогу можуть попросити допомогти з вирішенням проблем. Наприклад, для відновлення приміщення попросити певні матеріали. А інші люди можуть відгукнутися на таке прохання. Все це має бути враховано в роботі та реалізовано у вигляді відповідного програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз програмних засобів для благодійної допомоги

Звичайно, що на цей момент в Україна всім відомі вебсайти окремих благодійних фондів, які мають широку відомість у всій країні. Таким прикладом є зокрема вебсайт благодійного фонду «Повернись живим» [4] (рис. 1.1).

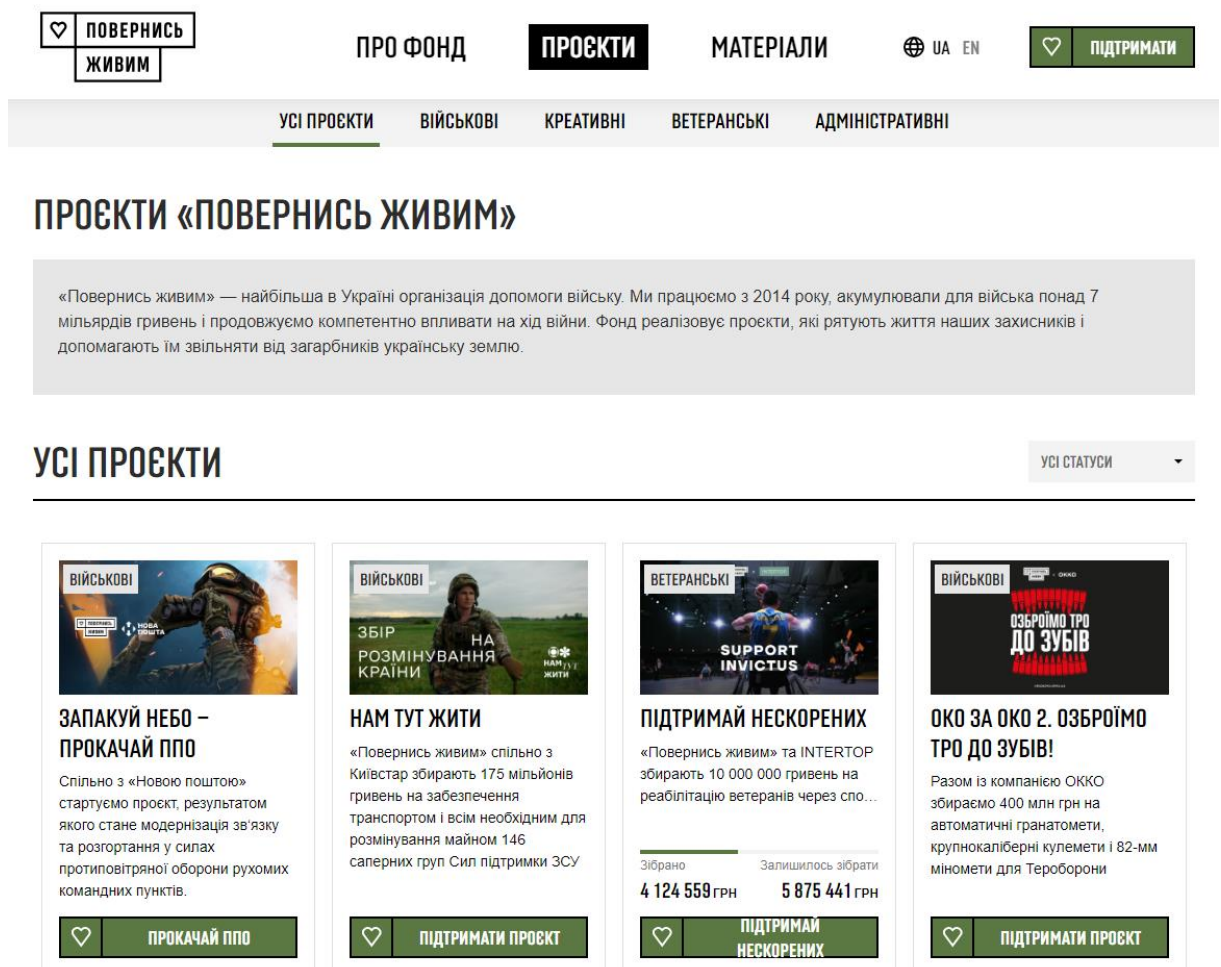


Рисунок 1.1 – Вебсайт благодійного фонду «Повернись живим»

Цей вебсайт має окремий розділ з проєктами, який виділений у відповідний пункт меню. Це дозволяє отримати інформацію про всі благодійні проєкти, які розпочаті на поточний момент і збір коштів на які виконується. Одразу біля кожного такого проєкту є посилання для того, що

перейти до пожертвування коштів. Звичайно, що це є дуже важливими для подібних вебсайтів. Підтримати можна і фонд загалом, і окремі проєкти. За кожним благодійним проєктом надається окрема інформація про очікувані результати, тобто не просто створюється опис проєкту, але і приводяться окремо ті закупки, які має бути в підсумку виконано. Проєкти також розбиті на групи, можна відповідно обирати потрібну групу і передивлятися проєкти тільки з неї, в тому числі ті, збір коштів на які вже завершився. Вебсайт також має розділ з матеріалами, де публікуються певні новини або загальні матеріали стосовно проєктів фонду або загалом певних ситуацій, з якими пов'язана діяльність фонду. Звичайно, що певною особливістю сучасних великих українських благодійних фондів є те, що вони фактично займаються військовою, а не гуманітарною допомогою.

Вебсайт благодійного фонду Сергія Притули [5] (рис. 1.2) має дещо іншу структуру роботи.

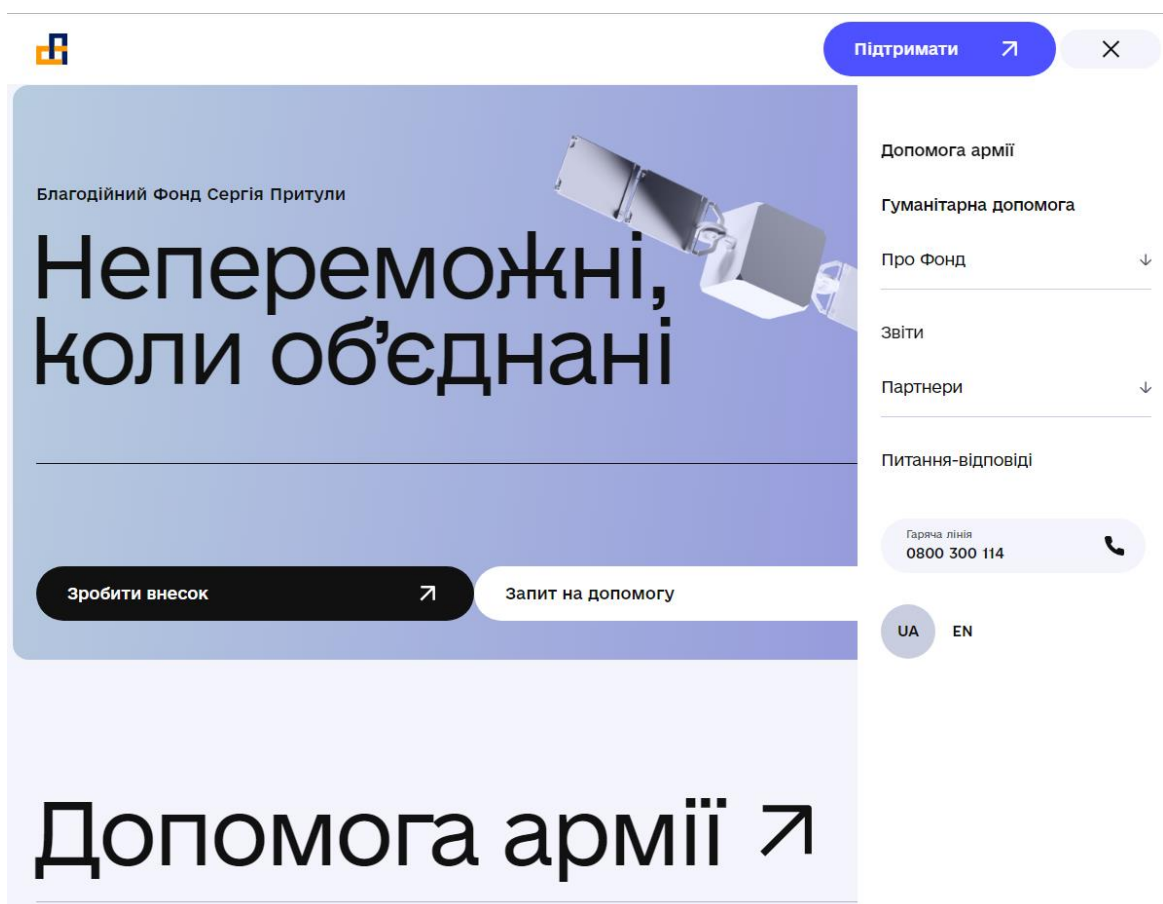


Рисунок 1.2 – Вебсайт благодійного фонду Сергія Притули

Цей вебсайт одразу на першій сторінці має посилання на пожертвування, але і на запит на допомогу. Тобто при необхідності отримати певні засоби можна попросити це від фонду, тобто таким чином виконується активний зв'язок, коли ініціатором допомоги може бути сам прохач допомоги. У певних ситуаціях це може бути дуже корисно. Даний фонд займається в тому числі і гуманітарною допомогою, відповідно є форма, яка дозволяє надати всі необхідні дані про себе для отримання такої допомоги.

Вебсайт має також і розділ з благодійними проєктами. Окремо виділені активні та архівні проєкти.

Однак загалом це вебсайти, які несуть інформативний напрямок, окрім того підтримуючи певні додаткові засоби, але існують також інструменти, які дозволяють керувати певною спільнотою, яка утворюється навколо благодійної організації. При цьому можуть визначатися окремі події, проєкти, направляючи інформацію про них певним категоріям користувачів. Зокрема така логіка використовується в Hivebrite [6], де можуть створюватися різноманітні заходи, наприклад, благодійні події, а квитки на ці події розповсюджуватися між користувачами, відповідно виконується керування подіями, квитками, платежами, в тому числі виконання цих платежів, керування ними. Користувачі можуть запитувати про допомогу інших користувачів, тобто це дозволяє організувати певну волонтерську підтримку проєктами.

Звичайно, що це не всі існуючі програмні засоби в цій сфері. Саме ці було відібрано через те, що вони як раз орієнтовані на певне керування спільнотою щодо організації допомоги, однак існують програмні засоби, які дозволяють керувати документацією благодійного фонду, аналізувати статистику роботи, як наприклад, CharityTracker [7]. Однак це вже програмні засоби, направлені на сам благодійний фонд, а не на його комунікацію, запити на підтримку та організацію підтримки стосовно користувачів.

1.2 Визначення концепції розроблюваної програми

Зважаючи на більш широкі і чітко визначені різноманітні інструменти, саме вебсайт благодійного фонду Сергія Притули, а також додаток Hivebrite, який має особливу концепцію порівняно з інформативними вебсайтами, обрано для порівняння (табл. 1.1). У дипломній кваліфікаційній роботі об'єднано різні засоби, які надаються в цих окремих інструментах, також вирішено підтримувати горизонтальну взаємодію між користувачами, тобто щоб не тільки сама організація, але і окремих користувач міг попросити про допомогу, а інші звичайні користувачі могли йому допомогти.

Таблиця 1.1 – Результати аналізу програмних засобів для благодійної допомоги

Ознака	Фонд Сергія Притули	Hivebrite	Дипломна робота
Інформування про благодійні проекти організації	+	+	+
Збір коштів на фінансування проектів	+	+	+
Підтримка керування продажем квитків на благодійні події	—	+	+
Запитування допомоги самими користувачами	+	—	+
Інформування про наявні благодійні пропозиції	—	Тільки стосовно роботи	+
Надання допомоги користувачам іншими користувачами	—	—	+

1.3 Висновки за розділом 1 та постановка завдань роботи

Виконавши аналіз предметної області, розглянувши аналоги, було вирішено розробити програмне забезпечення для благодійної допомоги, яке буде концентруватися не на окремій благодійній організації, а мати можливість представляти різні благодійні організації. Кожна така організація повинна мати цілий спектр інструментів, одна і звичайні користувачі повинні мати можливість горизонтальної взаємодії для допомоги один одному, коли в скрутній ситуації певними матеріалами, інструментом, працею один користувач може допомогти іншому. У підсумку завданнями роботи є:

- аналіз програмних аналогів і визначення концепції програми;
- проєктування програмного забезпечення для благодійної допомоги;
- реалізація вебдодатку для благодійної допомоги.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір технологій розробки

Технології розробки, а точніше веброзробки, на даний момент полягають в комплексі мови програмування та фреймворку, тому при порівнянні одразу це враховувалось, розглядаючи фреймворки Django і Spring (табл. 2.1). Вони дозволяють працювати з різними мовами програмування, відповідно Django [8] наслідує простоту і гнучкість мови Python [9], а Spring – продуктивність мови Java. Але при цьому всьому вони мають стосовно саме веброзробки також відмінності, адже Django має багато засобів з самого початку, які вже вбудовані і дозволяють швидше отримати результат, виконувати повнотекстовий пошук, якщо потрібно, тому в даному випадку це є перевагою. Тож за сукупністю ознак порівняння python і Django обрані в підсумку.

Таблиця 2.1 – Порівняння основних фреймворків веброзробки

Ознака	Django	Spring
Мова програмування	Python	Java
Архітектура	Model-view-template	Model-view-controller
Складність вивчення	Низька	Висока
Швидкість роботи	Середня	Висока
Пошук даних	Доступний в тому числі повнотекстовий	Повнотекстовий пошук можливий тільки з зовнішніми технологіями
Наявні засоби	Має багато вбудованих засобів, можуть встановлюватися додаткові модулі	В основному потребує встановлення додаткових модулів

2.2 Моделювання вимог до програми

Під час моделювання вимог до програми стосовно організації роботи звичайного користувача, який не має поточної авторизації в програмі, використано наступний перелік функціональних вимог (рис. 2.1):

- авторизація (альтернативним сценарієм є реєстрація);
- жертвування коштів на благодійний проєкт;
- жертвування коштів на благодійну організацію;
- придбання квитка на благодійну подію;
- перегляд сторінки благодійної організації;
- перегляд сторінки користувача;
- перегляд сторінки благодійного проєкту;
- перегляд сторінки благодійної події;
- перегляд сторінки прохання про допомогу;
- перегляд сторінки пропозиції потребуючим;
- пошук благодійних організацій;
- пошук благодійних проєктів;
- пошук благодійних подій;
- пошук прохань про допомогу;
- пошук пропозицій потребуючим;
- відтворення переліку благодійних проєктів організації;
- відтворення переліку діючих благодійних проєктів у місті;
- відтворення переліку діючих пропозицій потребуючим у місті;
- відтворення переліку благодійних подій організації;
- відтворення переліку пропозицій потребуючим організації;
- відтворення переліку прохань про допомогу користувача;
- відтворення переліку пожертвувань за благодійним проєктом;
- відтворення переліку груп квитків на благодійну подію;
- відтворення переліку скарг за благодійною організацією;
- відтворення переліку скарг за користувачем.

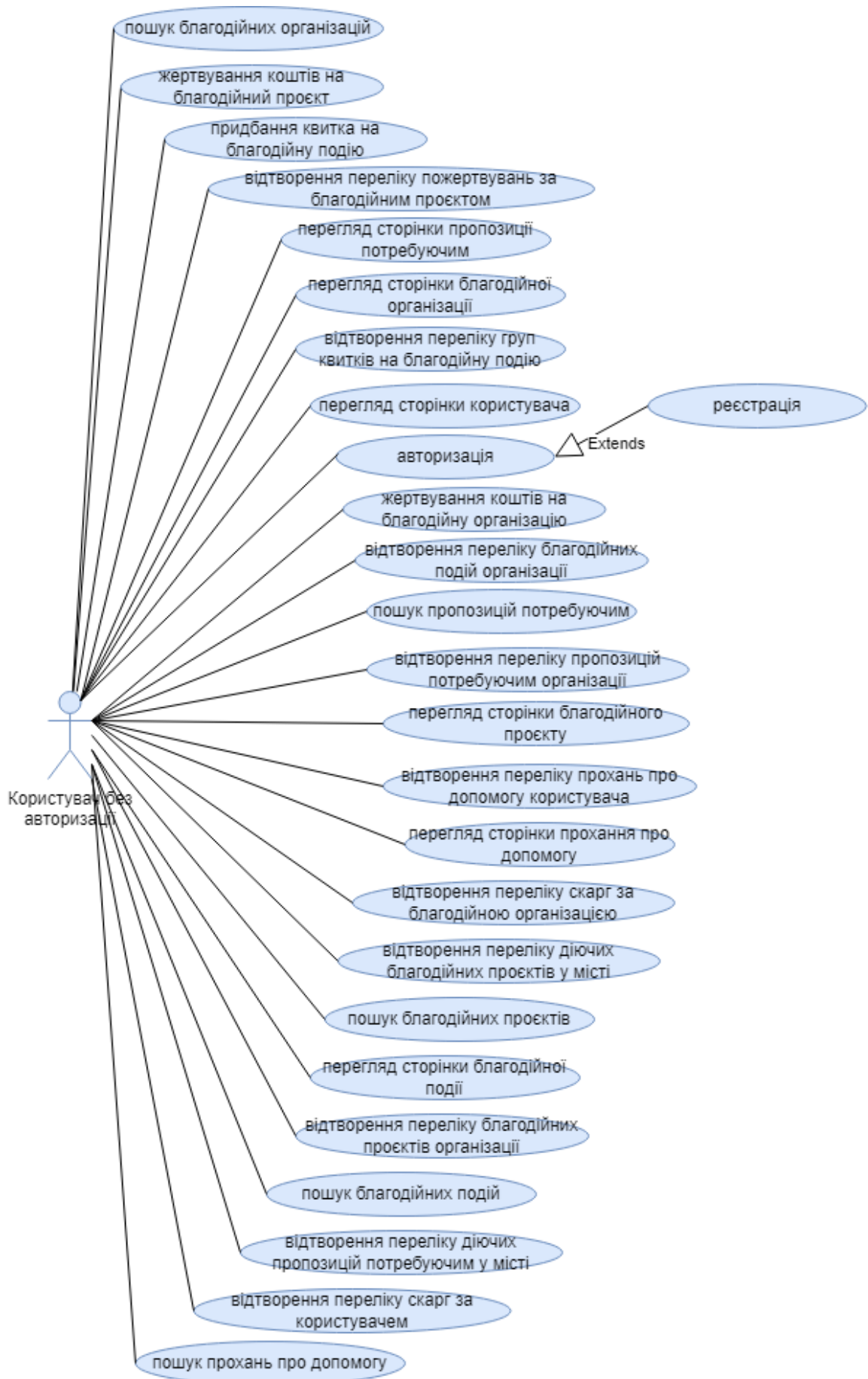


Рисунок 2.1 – Діаграма прецедентів користувача без авторизації

Під час моделювання вимог до програми стосовно організації роботи звичайного користувача, який вже виконав авторизацію в програмі, використано наступний перелік функціональних вимог (рис. 2.2):

- завантаження власних підтверджуючих документів;
- відтворення переліку власних підтверджуючих документів;
- вилучення власного документа;
- створення закладки на благодійну подію;
- створення прохання про допомогу;
- оновлення прохання про допомогу;
- створення коментаря до благодійного проєкту;
- створення коментаря до благодійної події;
- створення коментаря до благодійної організації;
- створення закладки на прохання про допомогу;
- створення закладки на пропозицію потребуєчим;
- подання запиту з пропозицією допомоги благодійному проєкту;
- подання запиту з пропозицією допомоги благодійній події;
- подання запиту з пропозицією допомоги благодійній організації;
- створення закладки на благодійний проєкт;
- відтворення переліку пожертвувачів користувача;
- відтворення переліку допомоги, отриманої користувачем;
- відтворення переліку власних придбаних квитків;
- вподобання благодійної події (альтернативний сценарій – скасування вподобання благодійної події);
- відтворення переліку закладок;
- вподобання пропозиції потребуєчим (альтернативний сценарій – скасування вподобання пропозиції потребуєчим);
- відтворення переліку вподобань;
- створення запиту на пропозицію потребуєчим;
- відтворення переліку власних запитів на пропозиції потребуєчим;
- подання скарги на благодійну організацію;

- відтворення власних створених скарг;
- вподобання благодійного проєкту (альтернативний сценарій – скасування вподобання благодійного проєкту);
- відтворення переліку власних запитів з пропозиціями допомоги;
- подання відповіді на прохання про допомогу;
- відтворення переліку власних відповідей на прохання про допомогу;
- відтворення отриманих відповідей на прохання про допомогу;
- прийняття отриманої відповіді на прохання про допомогу (альтернативний сценарій – відхилення отриманої відповіді на прохання).

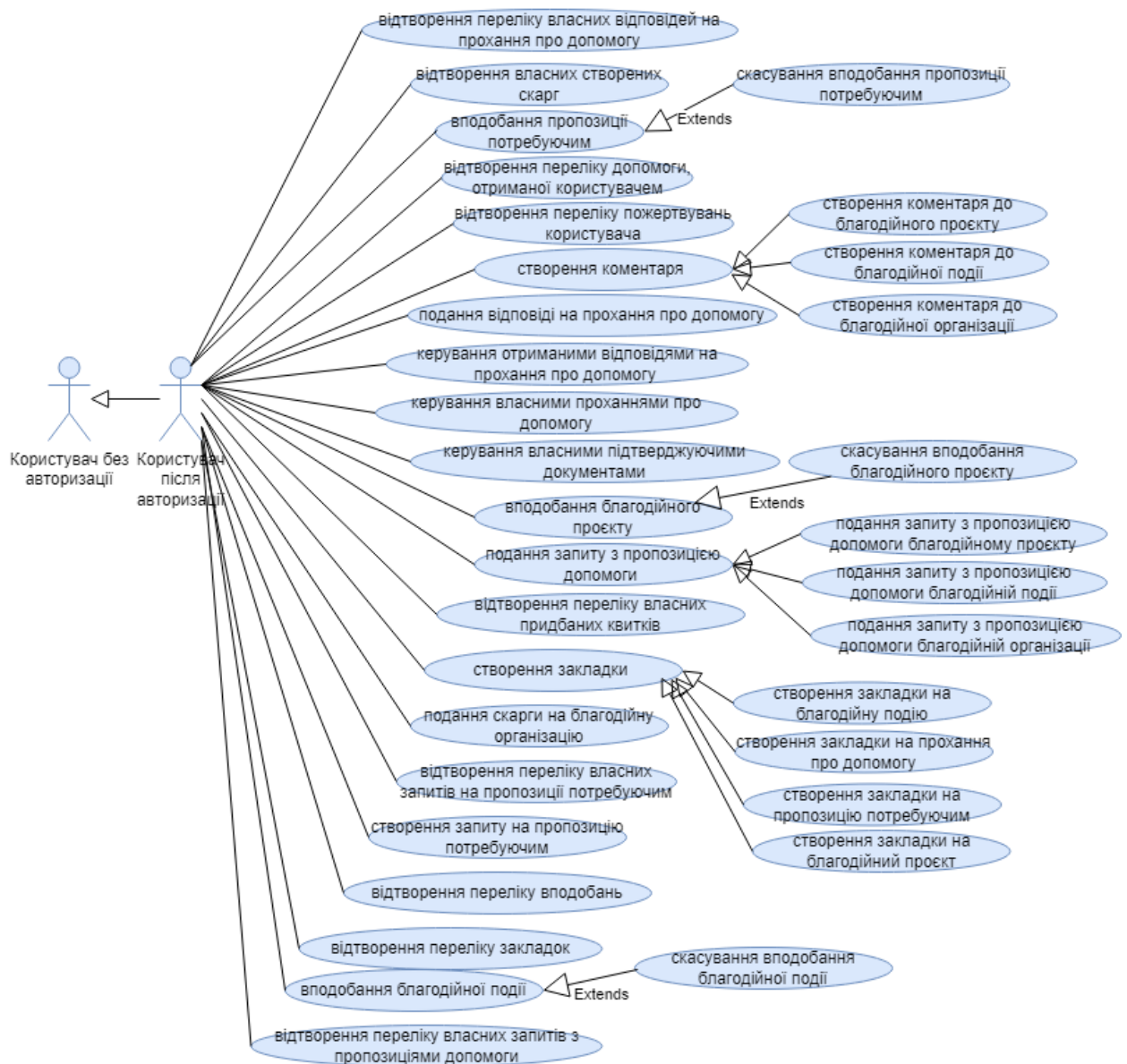


Рисунок 2.2 – Діаграма прецедентів користувача після авторизації

Під час моделювання вимог до програми стосовно організації роботи користувача, який є працівником благодійної організації, використано наступний перелік функціональних вимог (рис. 2.3):

- визначення користувача як працівника благодійної організації;
- оновлення даних благодійної організації;
- визначення нового благодійного проєкту організації;
- оновлення даних благодійного проєкту;
- закриття збору пожертвувань за благодійним проєктом;
- визначення нової благодійної події з визначенням доступних квитків;
- оновлення даних благодійної події та доступних квитків;
- визначення нової групи квитків на благодійну подію;
- оновлення даних групи квитків на благодійну подію;
- визначення нової пропозиції потребуєчим;
- оновлення даних пропозиції потребуєчим;
- додавання елементів до пропозиції;
- оновлення даних окремих елементів пропозиції потребуєчим;
- прийняття запиту з пропозицією допомоги благодійному проєкту або його відхилення;
- прийняття або відхилення запиту на пропозицію потребуєчим;
- відтворення переліку запитів на пропозицію потребуєчим;
- подання скарги на користувача;
- відтворення переліку запитів з пропозиціями допомоги за благодійним проєктом;
- відтворення переліку запитів з пропозиціями допомоги за благодійною подією;
- відтворення переліку запитів з пропозиціями допомоги за благодійною організацією;
- прийняття запиту з пропозицією допомоги благодійній події або його відхилення;

– прийняття запиту з пропозицією допомоги благодійній організації або його відхилення.



Рисунок 2.3 – Діаграма прецедентів працівника благодійної організації

2.3 Проектування бази даних

Забезпечено роботу з базою даних (БД) у цій роботі через систему PostgreSQL, яка є потужною і одночасно безкоштовною.

Усіх користувачів при проектуванні БД було представлено як єдину сутність, поле `role` якої визначає, чи це працівник однієї з благодійних організацій, чи це просто звичайний користувач програми, який є певним кандидатом на благодійну допомогу. Узагальнена сутність користувача представлена таблицею `Applicant`, до складу якої входять такі поля:

- `id` – ідентифікатор користувача, який є працівником благодійного фонду або кандидатом на благодійну допомогу;
- `name` – ім'я користувача;
- `login` – логін користувача;
- `picture` – зображення або фотографія, якою себе визначає в програмі користувач;
- `job` – род діяльності користувача (робота), якщо він працює;

- age – вік користувача;
- sex – стать;
- phone – номер телефону;
- email – електронна пошта;
- password – пароль;
- role – роль, яку виконує в програмі користувач: стандартна (користувач є кандидатом на благодійну допомогу) або розширена (користувач є працівником благодійної організації).

Для отримання деякої допомоги користувач може завантажувати підтверджуючі документи. Щоб не завантажувати їх на кожну заявку повторно, вони зберігаються окремо за користувачем, а потім додається тільки посилання на дані цього документу. Дані документів користувача знаходяться в таблиці ApplicantDocument, до складу якої входять такі поля:

- id – ідентифікатор документа користувача;
- applicant – користувач, який завантажив цей документ, є зовнішнім ключем з таблиці Applicant;
- document – документ, який було завантажено (фактично файл документа).

Благодійні організації знаходяться в таблиці CharityGroup, до складу якої входять такі поля:

- id – ідентифікатор благодійної організації;
- title – назва благодійної організації;
- description – опис благодійної організації;
- slug – унікальний ідентифікатор, остання частина Uniform Resource Locator (URL)-адреси сторінки благодійної організації;
- account – банківський рахунок, на який відбувається зарахування коштів при створенні добровільних пожертвувань цій благодійній організації;
- city – місто, де головний офіс благодійної організації, є зовнішнім ключем з таблиці City;

- address – адреса головного офісу;
- donated – зібрана на даний момент сума коштів;
- website – вебсайт благодійної організації;
- phone – номер телефону для контактування з благодійною організацією напряму поза межами програми;
- email – електронна пошта благодійної організації.

Благодійні організації в програмі визначаються працівниками, які від їх імені виконують певні функції з керування, наповнення даними. Надані права користувачам на роботу з благодійними організаціями знаходяться в таблиці GroupChariter, до складу якої входять такі поля:

- group – благодійна організація, в якій працює цей користувач, є зовнішнім ключем з таблиці CharityGroup;
- chariter – працівник, який працює в цій благодійній організації, маючи відповідні права для роботи з даними організації в програмі, є зовнішнім ключем з таблиці Applicant.

Благодійні проекти, які створюються організаціями, знаходяться в таблиці Charityproject, до складу якої входять такі поля:

- id – ідентифікатор благодійного проекту;
- sphere – сфера діяльності, до якої відноситься цей проект, є зовнішнім ключем з таблиці Sphere;
- charitygroup – основна благодійна організація, яка створила цей проект, є зовнішнім ключем з таблиці CharityGroup;
- title – назва благодійного проекту;
- core – повний опис суті благодійного проекту;
- slug – унікальний ідентифікатор, остання частина URL-адреси сторінки благодійного проекту;
- picture – зображення або фотографія, якою презентується благодійний проект у програмі;
- account – банківський рахунок, на який відбувається зарахування коштів при створенні добровільних пожертвувань цьому благодійному

проєкту: визначено для того, щоб працювати у тих випадках, коли для окремих проєктів створюється окремий рахунок, у випадку, якщо один рахунок використовується для всіх внесків, то це поле може бути залишено порожнім;

- website – посилання на вебсайт благодійного проєкту;
- goalsum – сума пожертвувань, яку необхідно зібрати за цим благодійним проєктом;
- donated – отримана сума пожертвувань на даний момент за цим благодійним проєктом;
- toend – визначає, чи можна вносити платежі після того, як потрібну суму зібрано;
- ended – дата завершення збору, якщо збір коштів на цей благодійний проєкт завершено;
- active – визначає дату, коли має бути розпочато збір коштів за цим благодійним проєктом: це дозволяє у випадку, якщо проєкт має бути запущений у певний момент, зробити це, щоб він не був доступний одразу після створення;
- saved – час створення благодійного проєкту.

За благодійним проєктом можуть визначатися окремі результати, які мають бути досягнуті. Наприклад, якщо благодійний проєкт передбачає придбання певної техніки, то може бути визначено, яка саме техніка потрібна. Ці результати проєкту знаходяться в таблиці CharityprojectPosition, до складу якої входять такі поля:

- id – ідентифікатор потрібного результату проєкту;
- charityproject – благодійний проєкт, є зовнішнім ключем з таблиці Charityproject;
- title – назва цього результату (наприклад, міномет певної моделі);
- description – опис результату (наприклад, можуть вказуватися конкретні характеристики цього міномету);

– picture – зображення або фотографія, якою презентується цей результат.

Пожертвування знаходяться в таблиці Donation, до складу якої входять такі поля:

- id – ідентифікатор пожертвування;
- applicant – користувач, який виконав це пожертвування, є зовнішнім ключем з таблиці Applicant;
- account – банківський рахунок, з якого відбулося зарахування коштів за пожертвуванням;
- accountgetter – банківський рахунок, на який відбулося зарахування коштів за пожертвуванням: оскільки в певних ситуаціях рахунок може бути змінений, то ці дані потрібно зберегти для уточнення фінансових деталей у майбутньому для вирішення будь-яких проблем;
- sum – сума пожертвування;
- saved – час створення пожертвування;
- paid – позначає факт того, що платіж виконано.

Платежі відповідно можуть стосуватися або благодійного проєкту, або благодійної організації загалом, або сплати за квиток на благодійну подію. Оскільки самі дані пожертвування при цьому однакові, а відрізняється тільки напрям фінансування (куди було надано кошти), то для цього створено зв'язуючі таблиці, кожна з яких з пожертвуванням зв'язана зв'язком один-до-одного, при чому одна сторона може дорівнювати і нулю. Тобто за кожним пожертвуванням встановлюється або благодійний проєкт, або благодійна організація загалом, або сплата за квиток на благодійну подію, але не декілька одночасно.

Відповідно зв'язки між благодійними проєктами і пожертвуваннями знаходяться в таблиці CharityprojectDonation, до складу якої входять такі поля:

- donation – пожертвування, яке було виконано за благодійним проєктом, є зовнішнім ключем з таблиці Donation;

– `charityproject` – благодійний проєкт, якого стосується пожертвування, є зовнішнім ключем з таблиці `Charityproject`.

Зв'язки між благодійними організаціями і пожертвуваннями знаходяться в таблиці `CharitygroupDonation`, до складу якої входять такі поля:

– `donation` – пожертвування, яке було виконано для благодійної організації, є зовнішнім ключем з таблиці `Donation`;

– `charitygroup` – благодійна організація, якої стосується пожертвування, є зовнішнім ключем з таблиці `CharityGroup`.

Зв'язки між квитками на благодійну подію і пожертвуваннями знаходяться в таблиці `CharityeventTicketDonation`, до складу якої входять такі поля:

– `donation` – пожертвування, яке було виконано для сплати за квиток на благодійну подію, є зовнішнім ключем з таблиці `Donation`;

– `charityeventticket` – квиток на благодійну подію, який було профінансовано цим пожертвуванням, є зовнішнім ключем з таблиці `CharityeventTicket`.

Благодійні події, на яких можуть збирати кошти, знаходяться в таблиці `Charityevent`, до складу якої входять такі поля:

– `id` – ідентифікатор благодійної події;

– `charitygroup` – благодійна організація, якою було створено цю благодійну подію, є зовнішнім ключем з таблиці `CharityGroup`;

– `title` – назва благодійної події;

– `description` – опис благодійної події;

– `city` – місто, де відбувається подія, є зовнішнім ключем з таблиці `City`;

– `slug` – унікальний ідентифікатор, остання частина URL-адреси сторінки благодійної події;

– `picture` – зображення або фотографія, якою презентується ця благодійна подія;

– `eventstart` – дата і час, коли має розпочатися ця благодійна подія;

- eventlong – тривалість події у хвилинах;
- eventplace – місце, де заплановано проведення події;
- website – посилання на вебсайт благодійної події;
- goalsum – сума, яку потрібно зібрати на цій благодійній події: може включати і вартість квитків, і додаткові збори під час події;
- donated – сума, яку вже було зібрано: до події враховуються придбані квитки;
- active – визначає дату, коли має бути розпочато продаж квитків на цю благодійну подію: це дозволяє у випадку, якщо продажу квитків має бути запущений у певний момент, зробити це, щоб інформація не була доступною одразу після створення;
- saved – дата і час створення запису про цю благодійну подію.

Вподобання благодійних подій користувачами вказують зокрема те, на скільки аудиторії подобається ідея події, знаходяться в таблиці CharityeventLike, до складу якої входять такі поля:

- charityevent – благодійна подія, яка сподобалась користувачу, є зовнішнім ключем з таблиці Charityevent;
- applicant – користувач, який вповодив цю благодійну подію, є зовнішнім ключем з таблиці Applicant.

Користувач може не тільки визначити, що йому подія сподобалась, але і створити закладку, щоб повернутися до події пізніше. Наприклад, якщо він хоче пізніше надати кошти або детальніше ознайомитися з подією. Закладки благодійних подій знаходяться в таблиці CharityeventSave, до складу якої входять такі поля:

- charityevent – благодійна подія, закладку на яку було створено, є зовнішнім ключем з таблиці Charityevent;
- applicant – користувач, який створив цю закладку на благодійну подію, є зовнішнім ключем з таблиці Applicant.

Благодійні події можуть організовуватися для збору коштів на місці, а можуть включати і продаж квитків для збору коштів за участь у події. Дані

про доступні квитки на благодійну подію знаходяться в таблиці CharityeventTicket, до складу якої входять такі поля:

- id – ідентифікатор групи квитків на благодійну подію;
- charityevent – благодійна подія, на яку продаються ці квитки, є зовнішнім ключем з таблиці Charityevent;
- price – вартість квитків цієї групи;
- number – кількість квитків цієї цінової групи;
- description – опис особливостей участі у події при придбанні квитків з цієї цінової групи.

Окрім збору коштів благодійні організації ще можуть надавати тим, хто цього потребує, певну допомогу, яка може включати речі, їжу або необхідні засоби будівництва. Відповідно такі пропозиції не потребують сплати, але вони визначають певний набір одягу, їжі тощо. Користувачі ж можуть надавати заявки на те, що вони б хотіли отримати таку допомогу.

Пропозиції потребуючим з благодійної (гуманітарної) допомоги організацій знаходяться в таблиці Offer, до складу якої входять такі поля:

- id – ідентифікатор пропозиції з благодійної (гуманітарної) допомоги організацій;
- charitygroup – благодійна організація, якою було створено цю пропозицію, яка може надати ці речі або інші предмети потребуючим, є зовнішнім ключем з таблиці CharityGroup;
- title – загальна назва пропозиції;
- address – адреса, де реалізується пропозиція;
- city – місто, де реалізується пропозиція;
- slug – унікальний ідентифікатор, остання частина URL-адреси сторінки пропозиції потребуючим від благодійної організації;
- requirements – вимоги, які потрібно виконати, щоб отримати речі (або інші елементи) з цієї пропозиції: можуть визначатися документи, які потрібно надати;

- active – визначає те, чи активна пропозиція зараз, чи вже все було віддано потребуючим;

- photo – фотографія або зображення, яке може презентувати цю пропозицію потребуючим;

- saved – дата і час створення пропозиції потребуючим.

Вподобання пропозицій потребуючим від благодійної організації користувачами вказують зокрема те, на скільки аудиторії подобається ідея події, знаходяться в таблиці OfferLike, до складу якої входять такі поля:

- offer – пропозиція потребуючим від благодійної організації, яка сподобалась користувачу, є зовнішнім ключем з таблиці Offer;

- applicant – користувач, який вповодобав цю пропозицію потребуючим від благодійної організації, є зовнішнім ключем з таблиці Applicant.

Користувач може не тільки визначити, що йому пропозиція сподобалась, але і створити закладку, щоб повернутися до пропозиції пізніше. Закладки пропозицій потребуючим від благодійної організації знаходяться в таблиці OfferSave, до складу якої входять такі поля:

- offer – пропозиція потребуючим від благодійної організації, закладку на яку було створено, є зовнішнім ключем з таблиці Offer;

- applicant – користувач, який створив цю закладку на пропозицію, є зовнішнім ключем з таблиці Applicant.

Елементи (речі) пропозиції знаходяться в таблиці Offereditem, до складу якої входять такі поля:

- id – ідентифікатор елемента пропозиції;

- offer – пропозиція потребуючим від благодійної організації, до якої належить цей елемент, є зовнішнім ключем з таблиці Offer;

- itemsubcategory – підкатегорія, до якої належить цей елемент, є зовнішнім ключем з таблиці Itemsubcategory;

- title – назва елемента пропозиції;

- photo – зображення або фотографія, яка презентує цей елемент пропозиції;

- quantity – кількість одиниць у цьому елементі (наприклад, кількість футболок вказаного розміру);
- description – повний опис елемента пропозиції, що визначає суть (якісь особливості речей, наприклад);
- active – визначає, чи цей елемент ще досі доступний.

Користувачі можуть створювати запити, щоб продемонструвати потребу в конкретних елементах пропозиції потребуючим від благодійної організації. Ці запити знаходяться в таблиці Offerrequest, до складу якої входять такі поля:

- id – ідентифікатор запиту на деякий елемент пропозиції;
- offereditem – елемент пропозиції, за яким створено цей запит, є зовнішнім ключем з таблиці Offereditem;
- number – кількість одиниць цього елемента пропозиції, яку потребує цей користувач;
- applicant – користувач, який створив цей запит, є зовнішнім ключем з таблиці Applicant;
- saved – дата і час, коли було створено цей запит користувачем;
- request – текст, який визначає потребу або певні особливості, що бажає надати кандидат на допомогу;
- active – стан запиту, який може бути або поданим, або затвердженим, або скасованим.

Зв'язки між документами користувачів і запитами, за якими вони ці документи подають для надання даних для пропозиції, знаходяться в таблиці OfferrequestDocument, до складу якої входять такі поля:

- offerrequest – запиту на елемент пропозиції від кандидата на допомогу, є зовнішнім ключем з таблиці Offerrequest;
- applicantdocument – документ користувача, який додано до запиту, є зовнішнім ключем з таблиці ApplicantDocument.

Категорії елементів пропозицій з допомоги, тобто фактично речей (одяг, їжа, матеріали), знаходяться в таблиці `Itemcategory`, до складу якої входять такі поля:

- `id` – ідентифікатор категорії речей;
- `title` – назва категорії речей.

Категорії складаються з підкатегорій (наприклад, футболки, сорочки, овочі), які знаходяться в таблиці `Itemsubcategory`, до складу якої входять такі поля:

- `id` – ідентифікатор підкатегорії;
- `itemcategory` – категорія, до якої відноситься підкатегорія, є зовнішнім ключем з таблиці `Itemcategory`;
- `title` – назва підкатегорії речей.

Скарги при виникненні проблем знаходяться в таблиці `Complaint`, до складу якої входять такі поля:

- `id` – ідентифікатор скарги;
- `charitygroup` – благодійна організація, яка або поскаржилась на кандидата на допомогу за його неправомірну поведінку, або на яку поскаржився кандидат, є зовнішнім ключем з таблиці `CharityGroup`;
- `applicant` – кандидат на допомогу, який або поскаржився на благодійну організацію, або на якого поскаржилась благодійна організація за неправомірну поведінку, є зовнішнім ключем з таблиці `Applicant`;
- `standard` – істинне значення вказує, що це благодійна організація скаржиться на кандидата на допомогу, хибне значення вказує, що навпаки кандидат скаржиться на благодійну організацію;
- `saved` – дата і час, коли було створено скаргу;
- `severity` – тяжкість проблеми, яку було визначено адміністратором після розгляду цієї скарги;
- `core` – суть проблеми, яка виникла і через яку було створено скаргу.

Користувачі можуть у програмі створювати коментарі, які стосуються або пропозицій, або благодійних проєктів, або подій. У всіх цих випадках

дані коментаря однакові, а ось прив'язування коментаря різне, тому коментар винесено окремо в окрему таблицю, а створено зв'язки її з відповідними складовими. Коментарі знаходяться в таблиці Comment, до складу якої входять такі поля:

- id – ідентифікатор коментаря;
- applicant – користувач, який створив коментар, є зовнішнім ключем з таблиці Applicant;

- text – текст коментаря;
- saved – дата і час, коли було залишено коментар.

Зв'язок між коментарем і пропозицією потребуючим від благодійної організації знаходиться в таблиці OfferComment, до складу якої входять такі поля:

- offer – пропозиція потребуючим від благодійної організації, до якої відноситься цей коментар, є зовнішнім ключем з таблиці Offer;
- comment – коментар, який асоційовано з цією пропозицією, є зовнішнім ключем з таблиці Comment.

Зв'язок між благодійною подією і коментарем знаходиться в таблиці CharityeventComment, до складу якої входять такі поля:

- charityevent – благодійна подія, до якої відноситься цей коментар, є зовнішнім ключем з таблиці Charityevent;
- comment – коментар, який асоційовано з цією благодійною подією, є зовнішнім ключем з таблиці Comment.

Зв'язок між коментарем і благодійним проектом знаходиться в таблиці CharityprojectComment, до складу якої входять такі поля:

- charityproject – благодійний проект, до якого відноситься цей коментар, є зовнішнім ключем з таблиці Charityproject;
- comment – коментар, який асоційовано з цим благодійним проектом, є зовнішнім ключем з таблиці Comment.

Користувачі програми можуть також самі пропонувати свою допомогу в організації проектів або іншій роботі. Наприклад, для того щоб

розвантажити допомогу може використовуватися робота тимчасових помічників. Відповідно кожен користувач, який бажає, може подати запит на таку допомогу. Запити можуть бути типовими, тому самі запити знаходяться в таблиці Volunteer, до складу якої входять такі поля:

- id – ідентифікатор запиту на власне виконання допомоги, з яким користувач звертається до благодійних організацій;
- applicant – користувач, який створив цей запит (фактично шаблон запиту), є зовнішнім ключем з таблиці Applicant;
- proposition – текст запиту, з яким звертається користувач.

Подання конкретних запитів з пропозицією допомоги благодійному проєкту користувач може виконувати самостійно, а ці дані знаходяться в таблиці CharityprojectVolunteer, до складу якої входять такі поля:

- charityproject – благодійний проєкт, є зовнішнім ключем з таблиці Charityproject;
- volunteer – шаблон запиту користувача, яким він звертається з запитом для допомоги благодійному проєкту, є зовнішнім ключем з таблиці Volunteer.

Подання конкретних запитів з пропозицією допомоги благодійній події користувач може виконувати самостійно, а ці дані знаходяться в таблиці CharityeventVolunteer, до складу якої входять такі поля:

- charityevent – благодійна подія, є зовнішнім ключем з таблиці Charityevent;
- volunteer – шаблон запиту користувача, яким він звертається з запитом для допомоги благодійній події, є зовнішнім ключем з таблиці Volunteer.

Подання конкретних запитів з пропозицією допомоги благодійній організації загалом користувач може виконувати самостійно, а ці дані знаходяться в таблиці CharitygroupVolunteer, до складу якої входять такі поля:

– charitygroup – благодійна організація, є зовнішнім ключем з таблиці CharityGroup;

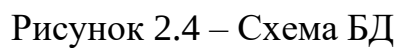
– volunteer – шаблон запиту користувача, яким він звертається з запитом для допомоги благодійній організації, є зовнішнім ключем з таблиці Volunteer.

Сфери діяльності, до яких належить благодійні проєкти, знаходяться в таблиці Sphere, до складу якої входять такі поля:

- id – ідентифікатор сфери діяльності;
- title – назва сфери діяльності.

Користувачі як кандидати на допомогу можуть створювати і власні прохання. Наприклад, якщо користувач потребує для ремонту в складних обставинах певних матеріалів, він може попросити про це. А благодійна організація або будь-який інший користувач може вирішити допомогти. Прохання користувачі про допомогу знаходяться в таблиці ApplicantAsk, до складу якої входять такі поля:

- id – ідентифікатор прохання користувача про допомогу;
- sphere – сфера діяльності, до якої відноситься це прохання (наприклад, будівництво), є зовнішнім ключем з таблиці Sphere;
- applicant – користувач, який створив це прохання, є зовнішнім ключем з таблиці Applicant;
- title – загальна назва оголошення про прохання;
- core – суть прохання;
- city – місто, де ця проблема виникла, є зовнішнім ключем з таблиці City;
- slug – унікальний ідентифікатор, остання частина URL-адреси сторінки прохання;
- picture – зображення або фотографія, яка асоційована з цим проханням;
- active – стан прохання: воно активне чи вже виконане, чи закрито невиконаним;



Відповіді на прохання користувача від інших користувачів знаходяться в таблиці ApplicantAskAnswer, до складу якої входять такі поля:

- id – ідентифікатор відповіді на прохання користувача;
- applicantask – прохання користувача, на яке цю відповідь надано, є зовнішнім ключем з таблиці ApplicantAsk;
- applicant – користувач, який створив цю відповідь, є зовнішнім ключем з таблиці Applicant;
- answer – текст відповіді;
- active – стан відповіді: активна або не розглянута адресатом, скасована або підтверджена;
- saved – дата і час створення відповіді на прохання користувача.

Закладки прохань про допомогу від користувачів можуть також створюватися іншими користувачами, знаходяться в таблиці ApplicantAskSave, до складу якої входять такі поля:

- id – ідентифікатор закладки;
- applicantask – прохання користувача, на яке було створено закладку, є зовнішнім ключем з таблиці ApplicantAsk;
- applicant – користувач, який створив цю закладку, є зовнішнім ключем з таблиці Applicant.

2.4 Висновки за розділом 2

Набір проєктних рішень у підсумку включив: мову програмування Python і систему керування БД PostgreSQL як засоби проєктування і розробки, моделювання вимог до програми через діаграми прецедентів для користувачів-кандидатів, які претендують на благодійну допомогу, та для працівників благодійних організацій, результати проєктування БД.

3 РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1 Схема функціонування програми

Схема функціонування програми (рис. 3.1) базується на меню роботи користувача і залежить від прав користувача та виконання або невиконання авторизації.

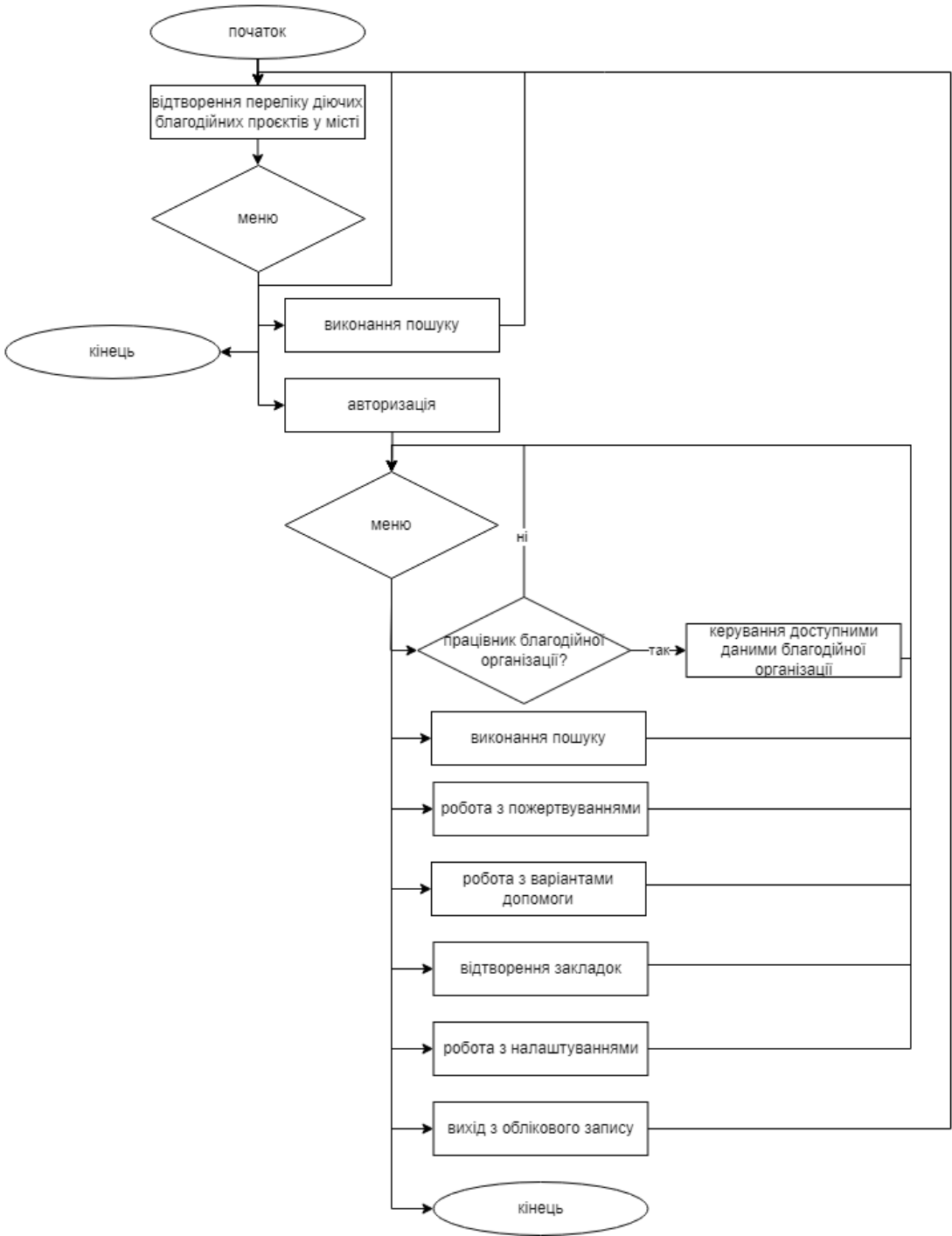


Рисунок 3.1 – Схема функціонування програми

3.2 Програмна реалізація функцій вебсистеми

Реалізацію функцій вебсистеми було виконано мовою програмування python з підключенням до системи для роботи з БД PostgreSQL через застосування фреймворку Django. Було використано підхід до реалізації представлень на основі функцій. Кожна програмна функція відповідає функціональній вимозі з врахуванням тих ролей користувачів, які можуть нею скористатися. Тому нижче збережено цю логіку опису: наведено перелік представлень і шаблонів вебсторінок за основними функціями програми. Для підтримки роботи цих функцій створено класи моделей, що повторюють структуру БД, а точніше її визначають.

Функція `charitygroup_projects` виконує відтворення переліку благодійних проєктів організації, яка має ідентифікатор (за полем `slug` моделі) `charitygroupSlug`. При виборі проєктів з БД враховується, що вони мають бути вже розпочаті, тому перевіряється те, що дата активності (поле `active`) для них містить значення, що не є пізнішим за поточний час. Усі такі благодійні проєкти впорядковуються за датою створення. А далі забезпечується для кожного такого благодійного проєкту обчислення кількості вподобань, що за ними міститься в БД, перевірка того, чи вподобав цей благодійний проєкт користувач, який авторизувався і відповідні дані авторизації якого містяться в параметрі `request` через `request.user`. Також визначається, чи створив за цим благодійним проєктом закладку користувач, який авторизувався. Така сама логіка використовується у всіх функціях, які виводять благодійні проєкти, події, пропозиції з допомоги, тобто за всіма встановлюється, чи вподобав, чи створив закладку користувач. Після цього всі визначені описаним чином дані заповнюють даними шаблон вебсторінки `projects.html`.

Функція `city_projects` виконує відтворення переліку діючих благодійних проєктів у місті та загалом. Вона має параметр `cityId`, який встановлює номер міста, благодійні проєкти за яким потрібно відібрати.

Якщо значення пропущено, то відбираються всі благодійні проєкти. При цьому це тільки ті, у яких зібрана сума менше ніж запланована, а вони при цьому розпочались, тобто дата активації раніша ніж поточна. Зібрані так благодійні проєкти формують дані шаблону вебсторінки `cityprojects.html`.

Функція `city_offers` виконує відтворення переліку діючих пропозицій потребуючим у місті, номер якого встановлюється параметром функції `cityId`. Визначаються тільки ще активні пропозиції, тобто ті, для яких ще елементи (речі) не віддані повністю. Після перевірки кількості уподобань, фактів вподобання та створення закладки користувачем, який авторизувався, шаблон вебсторінки `cityoffers.html` заповнюється даними.

Представлення відтворення переліку благодійних подій організації забезпечено функцією `charitygroup_events`, яка має ідентифікатор (за полем `slug` моделі) `charitygroupSlug`. При виборі подій з БД враховується, що продаж квитків за ними має бути вже розпочато або відповідно відкрито доступ до події, якщо квитки не мають продаватися. Підтримується шаблоном `events.html`.

Представлення відтворення переліку пропозицій потребуючим організації забезпечено функцією `charitygroup_offers`, яка має ідентифікатор (за полем `slug` моделі) `charitygroupSlug`. Після відбирання всіх пропозицій і визначення вподобань та закладок дані передаються в шаблон вебсторінки `offers.html`.

Представлення відтворення переліку прохань про допомогу користувача забезпечено функцією `applicant_asks`. Параметр `applicantLogin` визначає логін користувача, за яким потрібно виводити перелік прохань. За проханнями перевіряється факт створення закладки і відповідне значення додається в поле `save` кожного прохання. Результати передаються в шаблон вебсторінки `asks.html`.

Представлення відтворення переліку власних підтверджуючих документів реалізовано функцією `applicant_documents`. Дані беруться за користувачем, який авторизувався, і передаються в шаблон `documents.html`.

Таким чином, якщо користувач неавторизований, він не може отримати доступ до цих персональних даних. Відповідно таким чином користувач не може отримати доступ і до не своїх персональних даних.

Відтворення переліку пожертвувань користувача також відноситься до роботи з персональними даними, тому функція `applicant_donations` не має параметрів, а відбір пожертвувань здійснюється за даними авторизації. За кожним пожертвуванням також встановлюється, на що саме це було пожертвування, а цей об'єкт (благодійний проєкт, організація, подія) повертається в атрибуті `goal` до шаблону вебсторінки `donations.html`.

Відтворення переліку пожертвувань за благодійним проєктом реалізовано функцією `charityproject_donations`. Вона має параметр `charityprojectSlug`, що визначає ідентифікатор за полем `slug` моделі. Дані заповнюють шаблон `projectdonations.html`. При цьому дані виводяться в обмеженому вигляді, тобто не надається, наприклад, повні дані платника, тому це також належить до забезпечення захисту персональних даних.

Відтворення переліку допомоги, отриманої користувачем, реалізовано функцією `account_aids`, яка має параметр `accountLogin`, який визначає логін користувача, за яким цю інформацію потрібно отримати, а параметр `mode` за нульового значення вказує на відбір всіх даних, а за іншого – тільки отриманої допомоги, тобто там, де було отримано зворотній зв'язок і затверджено взаємодію. Дані заповнюють шаблон вебсторінки `aids.html`.

Представлення відтворення переліку власних придбаних квитків виконується на основі функції `applicant_tickets`. Дані моделі `CharityeventTicketDonation` відбираються за тими пожертвуваннями з моделі `Donation`, які належать користувачу, що авторизувався.

Відтворення переліку закладок виконано функцією `applicant_saves`. Відбираються всі створені закладки, включаючи збережені користувачем, що авторизувався, дані з моделі `CharityprojectSave`, прохання про допомогу на основі моделі `ApplicantAskSave`, благодійні події `CharityeventSave`, пропозиції

про допомогу на основі моделі OfferSave. У результаті заповнюється шаблон вебсторінки `saves.html`.

Представлення відтворення переліку власних запитів на пропозиції потребує функцією `applicant_requests`. Результат вноситься у вебсторінку `requests.html` за відповідними даними користувача, який авторизувався.

Відтворення переліку запитів на пропозицію потребує функцією `offer_requests`, яка має параметр `offerSlug`, який визначає ідентифікатор пропозиції за полем `slug` моделі. Цей параметр може бути пропущений, але вказаний конкретний елемент пропозиції (наприклад, футболки) за номером `offereditemId`. В залежності від того, який параметр вказано, відбираються відповідні дані. Якщо користувач, який авторизувався, має розширену роль, тобто він є працівником благодійної організації, то тільки тоді до результатів включаються не тільки пропозиції, але і зворотні запити за ними. Результати передаються в шаблон `offerrequests.html`.

Відтворення переліку скарг за благодійною організацією реалізовано функцією `charitygroup_complaints`, параметр якої `charitygroupSlug` визначає ідентифікатор благодійної організації за полем `slug` моделі. Відбираються всі скарги створені користувачами про цю благодійну організацію. Якщо значення параметра не задано, то для користувача, який є працівником благодійної організації, відбираються такі скарги за всіма благодійними організаціями, в яких він працює.

Представлення відтворення переліку скарг за користувачем реалізовано функцією `applicant_complaints`. Параметр `applicantLogin` визначає логін користувача, скарги на якого потрібно передати в шаблон `appcomplaints.html`.

Відтворення власних створених скарг реалізовано функцією `own_applicant_complaints`. Дані щодо користувача, який авторизувався, заповнюють шаблон вебсторінки `owncomplaints.html`.

Відтворення переліку власних запитів з пропозиціями допомоги представлено функцією `own_volunteers`, яка відбирає всі запити на волонтерство користувача, який авторизувався, включаючи запити щодо благодійних проєктів, подій та організацій, та вносить їх у сторінку `volunteers.html`.

Представлення відтворення переліку запитів з пропозиціями допомоги за благодійним проєктом з ідентифікатором `projectSlug` за полем `slug` моделі реалізовано функцією `project_volunteers`, що вносить ці дані в шаблон `projectvolunteers.html`.

Представлення відтворення переліку запитів з пропозиціями допомоги за благодійною подією з ідентифікатором `charityeventSlug` за полем `slug` моделі реалізовано функцією `event_volunteers`, що вносить ці дані в шаблон `eventvolunteers.html`.

Представлення відтворення переліку запитів з пропозиціями допомоги за благодійною організацією з ідентифікатором `charitygroupSlug` за полем `slug` моделі реалізовано функцією `group_volunteers`, що вносить ці дані в шаблон `groupvolunteers.html`.

Відтворення отриманих відповідей на прохання про допомогу з ідентифікатором `applicanttaskSlug` за полем `slug` моделі реалізовано функцією `applicant_ask_answers`, що вносить ці дані в шаблон `askanswers.html`.

Відтворення переліку власних відповідей на прохання про допомогу виконується функцією `own_ask_answers`, яка вносить у шаблон `ownanswers.html` відповіді, які були створені користувачем, який авторизувався.

Перегляд сторінки благодійної організації реалізовано функцією `charity_group`, яка за ідентифікатором `charitygroupSlug` за полем `slug` моделі відбирає дані самої благодійної організації, потім формує перелік з 3 найновіших благодійних проєктів цієї організації, перевіряючи кількість уподобань, уподобання і закладки користувача, який авторизувався, перелік з 3 найновіших благодійних подій цієї організації, перевіряючи кількість

уподобань, уподобання і закладки користувача, який авторизувався, перелік з 3 останніх пропозицій про допомогу цієї благодійної організації, перевіряючи кількість уподобань, уподобання і закладки користувача, який авторизувався. Усі ці результати передаються в шаблон сторінки `charitygroup.html`.

Перегляд сторінки благодійного проєкту реалізовано функцією `charity_project`, яка за ідентифікатором `charityprojectSlug` за полем `slug` моделі відбирає дані самого благодійного проєкту, за ним відбирає конкретні результати, коментарі до нього та загальну інформацію (уподобання, закладки). Ці результати передаються в шаблон `charityproject.html`.

Перегляд сторінки користувача реалізовано функцією `applicant`, яка за логіном користувача `applicantLogin`, а якщо він не заданий, то за даними користувача, який авторизувався, відбирає дані самого користувача, його прохання про допомогу. Результати формують шаблон вебсторінки `applicant.html`.

Перегляд сторінки благодійної події реалізовано функцією `charity_event`, яка має параметр `charityeventSlug` з ідентифікатором благодійної події за полем `slug` моделі. Окрім витягання даних самої благодійної події витягаються також дані пропонованих квитків на цю подію, коментарі до події.

Перегляд сторінки прохання про допомогу реалізовано функцією `applicant_ask`, яка має параметр `applicanttaskSlug` з ідентифікатором прохання за полем `slug` моделі. Усі дані прохання витягаються і передаються в шаблон вебсторінки `applicanttask.html`.

Перегляд сторінки пропозиції потребуючим реалізовано функцією `offer`, яка має параметр `offerSlug` з ідентифікатором пропозиції за полем `slug` моделі. Усі дані пропозиції витягаються і передаються в шаблон вебсторінки `offer.html`, при цьому перевіряючи факти уподобання та створення закладки за пропозицією користувачем, який авторизувався.

Пошук благодійних організацій реалізовано функцією `charity_group_find`, яка має параметри `charityTitle`, який визначає повну або часткову назву благодійної організації, та `cityName`, який визначає назву міста. Підтримка роботи пошуку організована на основі шаблону `findcharities.html`.

Пошук благодійних проєктів реалізовано функцією `charity_project_find`, яка має параметри `projectTitle`, який визначає повну або часткову назву благодійного проєкту, `sphereTitle`, який визначає назву сфери діяльності, `minSum`, який визначає мінімальну суму, яку заплановано зібрати за благодійним проєктом, `maxSum`, який визначає максимальну суму, яку заплановано зібрати за благодійним проєктом, `donatedSum`, який визначає мінімальну суму, яку було пожертвовано на благодійний проєкт, `active`, який визначає стан збору на благодійний проєкт (0 – всі, 1 – тільки розпочаті, 2 – кошти успішно зібрані, 3 – кошти не зібрано), `periodStart`, який визначає початок періоду збору коштів на благодійний проєкт, `periodEnd`, який визначає кінець періоду збору коштів на благодійний проєкт, `cityName`, який визначає назву міста, `minLikes`, що визначає мінімальну кількість уподобань, та `minComments`, що визначає мінімальну кількість коментарів за благодійним проєктом. Підтримка роботи пошуку організована на основі шаблону `findprojects.html`.

Пошук благодійних подій реалізовано функцією `charity_events_find`, яка має параметри `charitygroupSlug`, який визначає ідентифікатор за полем `slug` благодійної організації, яка цю подію влаштовує, `eventTitle`, який визначає назву або частину назви події, `minSum`, який визначає мінімальну суму, яку заплановано зібрати за благодійною подією, `maxSum`, який визначає максимальну суму, яку заплановано зібрати за благодійною подією, `donatedSum`, який визначає мінімальну суму, яку було пожертвовано на благодійну подію, `active`, який визначає стан збору на благодійну подію (0 – всі, 1 – тільки розпочаті, 2 – кошти успішно зібрані, 3 – кошти не зібрано), `eventStart`, який визначає початок періоду, коли заплановано проведення

самої події, `eventEnd`, який визначає кінець періоду, коли може проводитися подія, `cityName`, який визначає назву міста, `minLikes`, що визначає мінімальну кількість уподобань, та `minComments`, що визначає мінімальну кількість коментарів за благодійною подією. Підтримка роботи пошуку організована на основі шаблону `findevents.html`.

Пошук прохань про допомогу реалізовано функцією `applicant_ask_find`, яка має параметри `askTitle`, який визначає повну або часткову назву прохання, `sphereTitle`, який визначає назву сфери діяльності, `cityName`, який визначає назву міста. Підтримка роботи пошуку організована на основі шаблону `findask.html`.

Відтворення переліку груп квитків на благодійну подію реалізовано функцією `event_tickets`, яка має параметр `eventSlug`, що визначає ідентифікатор події. Дані передаються в шаблон вебсторінки `eventtickets.html`.

Відтворення переліку вподобань реалізовано функцією `applicant_likes`, яка за авторизаційними даними вибирає всі уподобання цього користувача благодійних проєктів, подій, пропозицій та перевіряє, чи вони були збережені цим користувачем у вигляді закладок. Усі ці дані передаються в шаблон вебсторінки `likes.html`.

Вподобання пропозиції потребуючим та скасування його реалізовано функцією `like_offer`, яка має параметр `offerSlug`, що визначає ідентифікатор пропозиції за полем `slug`. Ця функція перевіряє, чи користувач, який авторизувався, вже вподобав цю пропозицію, скасовуючи її, якщо так, або створюючи її, якщо ні. Аналогічно влаштовано вподобання благодійного проєкту і скасування його за допомогою функції `like_charity_project` з параметром `charityprojectSlug`, та вподобання благодійної події і скасування його за допомогою функції `like_charity_event` з параметром `charityeventSlug`.

Прийняття запиту з пропозицією допомоги благодійному проєкту реалізовано функцією `confirm_volunteer_charity_project` з параметром `charityprojectvolunteerId`, що визначає номер відповідного запису про

допомогу. Функція змінює значення поля `active` відповідного об'єкта в БД. Не має шаблону, а виконує перенаправлення. Так само працюють і перелічені вище функції підтримки уподобання. Аналогічно влаштовано відхилення запиту функцією `cancel_volunteer_charity_project` з параметром `charityprojectvolunteerId`. Аналогічно на основі функцій `confirm_offer_request` `cancel_offer_request` влаштовано прийняття або відхилення запиту на пропозицію потребуючим.

Створення закладки на благодійний проєкт реалізовано функцією `charity_project_save`, яка має параметр `charityprojectSlug`, який визначає ідентифікатор благодійного проєкту, за яким створюється екземпляр моделі з цим проєктом і даними користувача, який авторизувався. Якщо такий об'єкт вже є, то екземпляр навпаки вилучається. Аналогічно влаштовані створення закладки на пропозицію потребуючим функцією `offer_save` з параметром `offerSlug`, створення закладки на прохання про допомогу функцією `ask_save` з параметром `askSlug`, створення закладки на благодійну подію функцією `charity_event_save` з параметром `eventSlug`.

Жертвування коштів на благодійний проєкт реалізовано функцією `charity_project_donate`, яка має параметри `charityprojectSlug`, який визначає ідентифікатор благодійного проєкту, та `sum`, який визначає суму перерахування. Аналогічно влаштовані функції `charity_group_donate` для жертвування коштів на благодійну організацію, `charity_event_ticket_donate` для придбання квитка на благодійну подію. Окрема функція `on_pay` контролює отримання результатів сплати і у відповідності від направленості платежу оновлює інформацію про сплату за пожертвуванням, оновлює суму зібраних коштів за благодійним проєктом або відповідно організацією чи подією, у останньому випадку фіксуючи зміни у кількості квитків у `CharityeventTicket`.

Вилучення власного документа підтримується функцією `delete_document`, яка має параметр `documentId`, що визначає номер документа,

та після перевірки того, що власником документа є користувач, який авторизувався, виконує вилучення з БД.

Закриття збору пожертвувань за благодійним проєктом реалізовано функцією `end_donate` з параметром `charityprojectSlug`, який визначає ідентифікатор благодійного проєкту. Окрім внесення змін у БД функція перевіряє, що користувач, який авторизувався, є працівником благодійної організації, яка створила цей проєкт.

Інші представлення організовані за подібною логікою. Вони пов'язані з визначенням нових об'єктів та оновленням існуючих. Робота кожного з них організована через створення відповідного класу форми. Клас форми базується на визначенні полів у відповідності з полями відповідної таблиці БД, що було представлено в розділі 2. Так визначення нового благодійного проєкту організації реалізовано функцією `create_charity_project`. Ця функція окрім створення благодійного проєкту дозволяє обрати користувачу ту благодійну організацію, яка створює проєкт, з тих, в яких він працює. Якщо така організація одна, то відповідно одразу вносить це як відповідне поле в БД і зберігає дані. Оновлення даних благодійного проєкту реалізовано функцією `update_charity_project` з параметром `charityprojectSlug`, який визначає ідентифікатор благодійного проєкту, дані якого оновлюються. Так само відбувається перевірка права власності, тобто що працівник працює в цій благодійній організації.

Усі шаблони вебсторінок створені з визначенням макетів на основі шаблону [10].

3.3 Висновки за розділом 3

У повній мірі ті функції, які визначені були в розділі 2, реалізовані у складі вебсистеми з підтримкою фреймворку Django, визначенням представлень на основі створення функцій, а моделей і форм як класів.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для підтримки роботи благодійників та благодійних організацій в частині отримання ресурсу для допомоги та знаходження потребуючих, щоби визначити, чи варто надавати їм допомогу, а також кому саме конкретну допомогу надавати. Благодійні організації можуть створювати благодійні проєкти, пропозиції проведення благодійних подій з продажем квитків на них за необхідності, визначити пропозиції для надання допомоги потребуючим. Звичайні користувачі можуть виконувати пошук відповідних благодійних організацій, проєктів, подій, пропозицій, створювати власні пропозиції допомогти в їхній реалізації, подавати заявки на отримання допомоги, надавати власні пожертвування, просити про допомогу інших користувачів для вирішення конкретних проблем, а ті, в свою чергу, можуть надавати таку підтримку.

4.2 Умови виконання програми

На сервері виконання програми можливе тільки за наявності інтерпретатора мови python, системи PostgreSQL, процесора у мінімальній конфігурації Intel Core 2 або відповідного еквівалента чи більш потужного, оперативної пам'яті з доступом не менше ніж до 4 Гб, жорсткого диску з бажаним доступним обсягом не менше ніж 100 Гб.

4.3 Виконання і тестування програми

Тестування здійснено виконанням програми за різних умов. Такі умови визначалися різними браузерами на стороні користувача. Конкретно було протестовано роботу програми за визначеними вимогами для Chrome та Firefox. Елементи інтерфейсу, визначені відповідними шаблонами сторінок, в

обох браузерів відображались конкретно, результати за виконання функцій були отримані також коректні.

Звичайно, що центральним елементом керування своєю роботою для звичайного користувача є розділ Пошук меню. Цей розділ містить відповідні підрозділи, що визначають пошук благодійних організацій, пошук благодійних проєктів, пошук благодійних подій, пошук прохань про допомогу, пошук пропозицій потребувачам. Після вибору відповідного підрозділу на сторінці відповідного пошуку також є кнопки для переходу між цими варіантами пошуку. Окрім того зверху є поле для пошуку, в яке можна ввести тільки назву благодійної організації, після відкриття результатів знову ж таким можна перемкнутися на потрібний варіант пошуку. В залежності від обраного варіанта відображаються відповідні параметри пошуку (рис. 4.1), як визначено в розділі 3.

Рисунок 4.1 – Визначення параметрів пошуку

Коли параметри задані, кнопка Шукати відтворює нижче перелік результатів (рис. 4.2). Звичайно, що результати виділяються, по-перше, варіантом пошуку. Кожен результати пошуку благодійних проєктів містить фотографію, кнопку для переходу одразу до пожертвування, суму, яку потрібно зібрати (згори), сферу діяльності (біля синього кола), назву проєкту, інформацію про благодійну організацію. На неї можна натиснути і відкрити сторінку благодійної організації. Так само можна натиснути на назву благодійного проєкту і відкрити його сторінку. Кожен проєкт також включає дані про міста, де він виконується, кількість уподобань, зібрану суму коштів (внизу), а також іконки для вподобання та створення закладки.

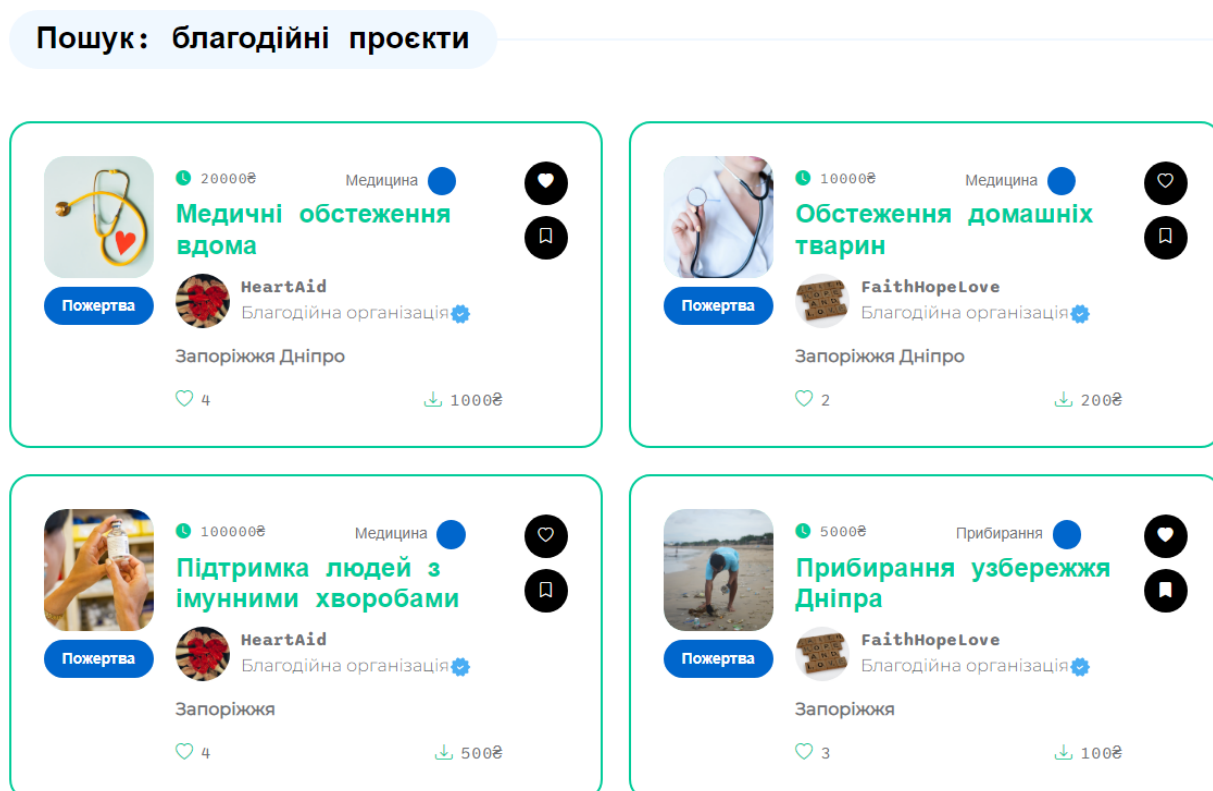


Рисунок 4.2 – Відображення результатів пошуку благодійних проєктів

Сторінка самого благодійного проєкту (рис. 4.3) містить його назву, потрібну для збору суму, кількість уподобань, іконки для вподобання і створення закладки, сферу діяльності, опис, кнопку для переходу до пожертвування, для подання запиту на волонтерство за цим проєктом.

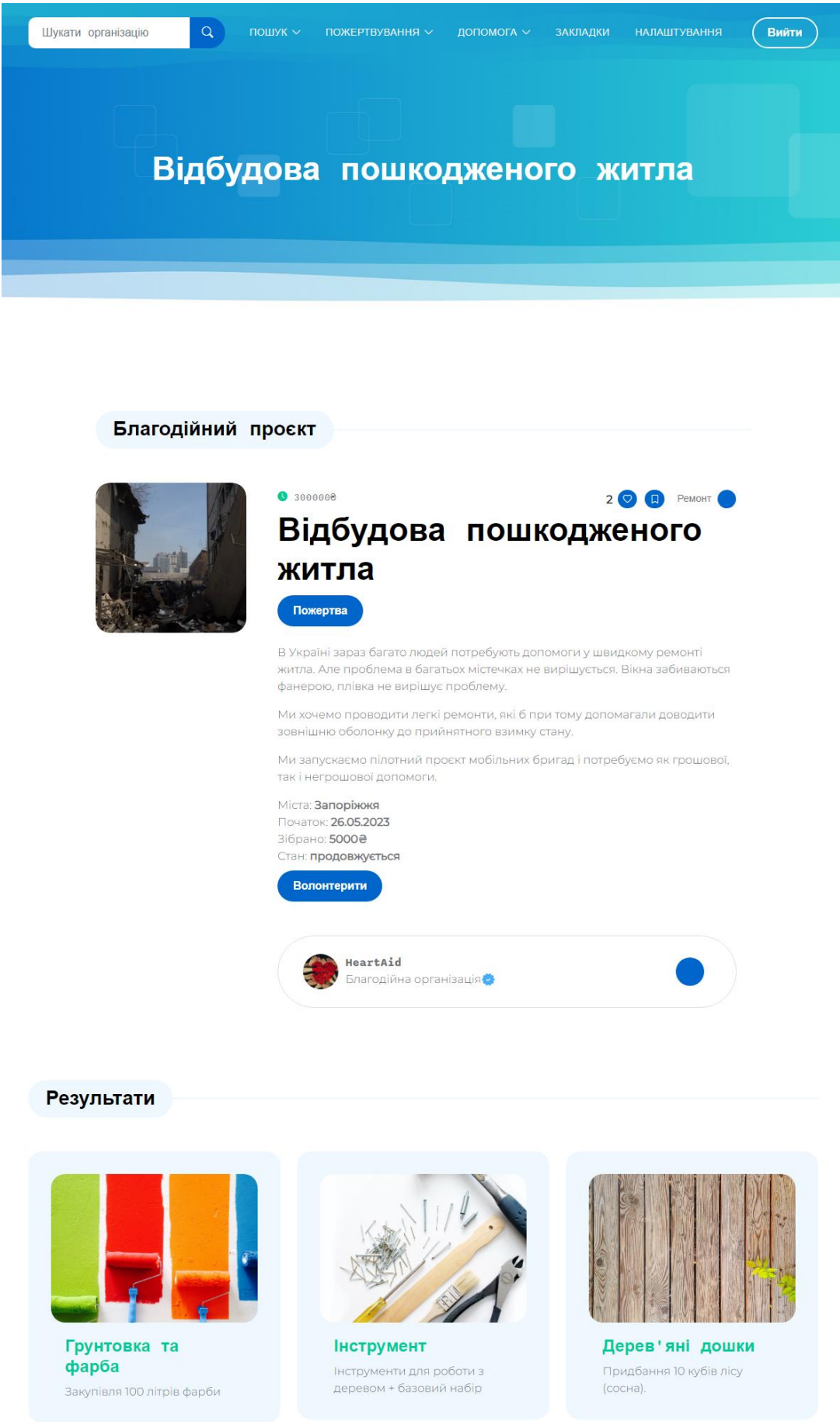


Рисунок 4.3 – Сторінка благодійного проєкту

Також за благодійним проєктом виводиться міста, де він реалізується, дата початку збору коштів за ним, зібрана сума, стан. Нижче знаходяться дані благодійної організації, перелік визначених результатів благодійного проєкту та коментарі до проєкту нижче (можна також і створити коментар, якщо користувач авторизований).

Фактично пошук благодійних організацій, проєктів, подій виконується, щоб знайти той варіант пожертвування власних коштів або волонтерства, який задовольняє користувача. Пошук пропозицій потрібен, щоб знайти потребуючому той одяг, наприклад, який йому потрібен, через благодійну допомогу. Також пошук може виконуватися і щодо прохань про допомогу. На сторінці прохання про допомогу (рис. 4.4) користувач має визначити стан, назву, опис, місто, де він знаходиться, фотографію, які переглядає кожен користувач, який думає йому допомогти. Прохання не можна уподобати. А тільки створити закладку, щоб повернутися.

Прохання про допомогу



активний

Ремонт

Допомога у відновленні житла

Відповісти

У будинку пошкоджено фасад, вибиті вікна. Потрібно терміново закрити вікна. Можливо хтось може допомогти зі склом, бо його важко знайти. Роботу можу зробити сам. А ось з матеріалами проблема.

Місто: Запоріжжя



Ігор Ностренко

Рисунок 4.4 – Сторінка прохання про допомогу користувача

Переглянути всі закладки, визначаючи на сторінці, закладки чого саме потрібно показати, можна з меню Закладки.

У випадку, якщо користувач є працівником благодійного фонду, то в нього в меню відображається підрозділ Керування, який дозволяє керувати благодійними проєктами, даними організації, пропозиціями, користувачами, додаючи на сторінці користувача до організації. Керування пропозиціями потребує (рис. 4.5) містить перелік таких пропозицій. За кожною пропозицією можна її або оновити, або перейти до переліку запитів за пропозицією. Сторінка також дозволяє створити нову пропозицію.

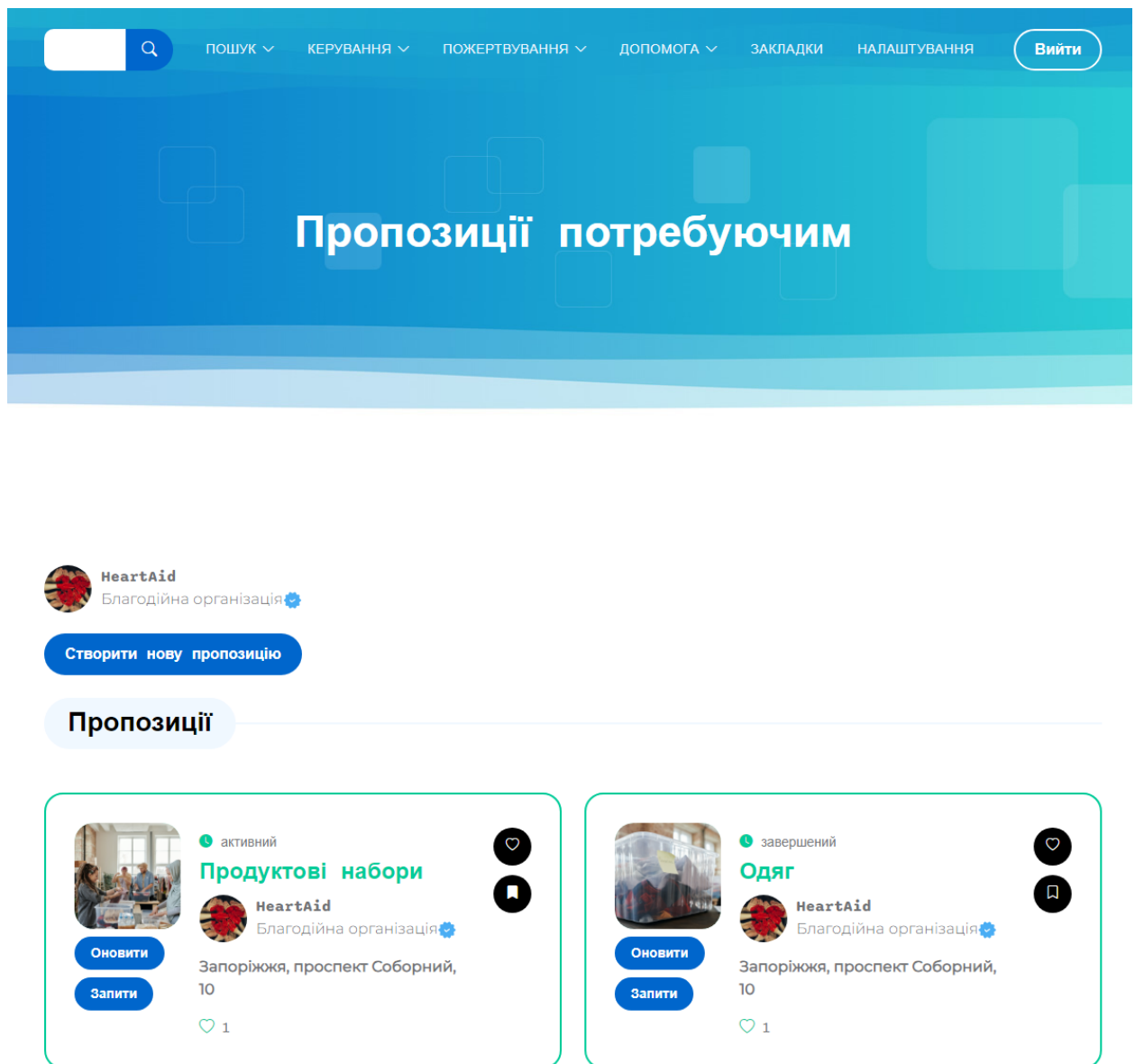


Рисунок 4.5 – Пропозиції потребує благодійної організації

Зі сторінки оновлення або зі сторінки самої пропозиції можна працівнику благодійної організації перейти до додавання елементів пропозиції. За цією ж логікою влаштовані всі підрозділи меню Керування.

Розділи меню Допомога і Пожертвування доступні всім користувачам після авторизації. Розділ Допомога містить самі прохання, які створив користувач, а відповідно з кожного прохання власного користувач може бачити і посилання для перегляду всіх, хто на них відповів. У цьому ж розділі знаходяться власні відповіді на прохання та запити щодо допомоги (про волонтерську фактично участь у благодійних проєктах, подіях). У розділі Пожертвування містяться самі пожертвування, які сплатив користувач, та квитки на благодійні події, які користувач придбав.

З розділу Налаштування доступні з самої сторінки дані про користувача, який авторизувався, також можливість подати скаргу (на відповідних сторінках кнопка знаходиться в самому низу), переглянути створені скарги, переглянути уподобання всі, які виконав користувач, що авторизувався (наявні кнопки для переходу між тим, уподобання чого саме потрібно відобразити), а також власні завантажені документи, які потім при поданні заявки на пропозицію допомоги можна під'єднати до форми.

4.4 Висновки за розділом 4

Логіка роботи користувача з програмою є достатньо простою, відповідає принципам, які використовуються у сучасних вебдодатках. І представникам благодійної організації, і звичайним користувачам надаються всі потрібні функції.

ВИСНОВКИ

Для забезпечення взаємодії благодійних організацій з людьми та між людьми на пряму створено програмне забезпечення в цій дипломній кваліфікаційній роботі.

Набір проєктних рішень щодо даного програмного забезпечення включив: мову програмування Python і систему керування БД PostgreSQL як засоби проєктування і розробки, моделювання вимог до програми через діаграми прецедентів для користувачів-кандидатів, які претендують на благодійну допомогу, та для працівників благодійних організацій, результати проєктування БД.

Розроблена програма призначена для підтримки роботи благодійників та благодійних організацій в частині отримання ресурсу для допомоги та знаходження потребуючих, щоби визначити, чи варто надавати їм допомогу, а також кому саме конкретну допомогу надавати. У повній мірі заплановані функції реалізовані у складі вебсистеми з підтримкою фреймворку Django, визначенням представлень на основі створення функцій, а моделей і форм як класів.

Використовуючи створене програмне забезпечення, благодійні організації можуть створювати благодійні проєкти, пропозиції проведення благодійних подій з продажем квитків на них за необхідності, визначити пропозиції для надання допомоги потребуючим, звичайні користувачі можуть виконувати пошук відповідних благодійних організацій, проєктів, подій, пропозицій, створювати власні пропозиції допомогти в їхній реалізації, подавати заявки на отримання допомоги, надавати власні пожертвування, просити про допомогу інших користувачів для вирішення конкретних проблем, а ті, в свою чергу, можуть надавати таку підтримку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Global \$525+ Bn NGOs and Charitable Organizations Markets Analysis, Opportunities and Strategies, 2015 – 2020, 2025F, 2030F [Electronic resource]. – Access mode : <https://www.globenewswire.com/en/news-release/2022/03/29/2411561/28124/en/Global-525-Bn-NGOs-and-Charitable-Organizations-Markets-Analysis-Opportunities-and-Strategies-2015-2020-2025F-2030F.html>.
2. Trust And Foundations Global Market Report 2023 [Electronic resource] : Increases in Philanthropy and Charity Activities Worldwide Bolsters Sector. – Access mode : <https://finance.yahoo.com/news/trust-foundations-global-market-report-111300105.html>.
3. NGOs And Charitable Organizations Market 2022 [Electronic resource]. – Access mode : <https://www.thebusinessresearchcompany.com/report/ngos-and-charitable-organizations-market>.
4. Головна сторінка – savelife.in.ua [Електрон. ресурс]. – Режим доступу : <https://savelife.in.ua/>.
5. Фонд Сергія Притули [Електрон. ресурс]. – Режим доступу : <https://prytulafoundation.org/>.
6. Hivebrite. The online community platform [Electronic resource]. – Access mode : <https://www.charitytracker.com/>.
7. CharityTracker [Electronic resource]. – Access mode : <https://info.charitytracker.com/non-profits?source=GetApp>.
8. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / Ben Shaw, Saurabh Badhwar, Andrew Bird et al. – Birmingham : Packt Publishing, 2021. – 826 p.
9. Python in a Nutshell [Text] : A Desktop Quick Reference / Alex Martelli, Anna Ravenscroft, Steve Holden et al. – Massachusetts : O'Reilly Media, 2023. – 735 p.
10. Pod Talk Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-584-pod-talk>.

ДОДАТОК А
Технічне завдання

Вступ

Благодійна допомога має різні сторони і охоплює звичайно не тільки підтримку людей під час війни. Збір коштів є важливим для реалізації будь-яких проєктів, тому це має бути враховано в роботі. Але окрім цього благодійна організація може надавати допомогу тим, хто її потребує, визначаючи гуманітарну допомогу і шукаючи тих, у кого дійсно є потреба. Окрім того самі кандидати на допомогу можуть попросити допомогти з вирішенням проблем. Наприклад, для відновлення приміщення попросити певні матеріали. А інші люди можуть відгукнутися на таке прохання. Все це має бути враховано в роботі.

A.1 Підстави для розробки

Завдання на дипломну кваліфікаційну роботу визначено на підставі наказу № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка» за темою «Програмне забезпечення для благодійної допомоги».

A.2 Призначення розробки

Програма призначена для підтримки роботи благодійників та благодійних організацій в частині отримання ресурсу для допомоги та знаходження потребуючих, щоби визначити, чи варто надавати їм допомогу, а також кому саме конкретну допомогу надавати.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для підтримки описаного призначення програма повинна надавати користувачам наступний набір функцій для роботи:

- авторизація всіх типів користувачів;
- самостійна реєстрація кандидатів на допомогу (звичайних користувачів);
- визначення користувача як працівника благодійної організації;
- завантаження власних підтверджуючих документів;
- оновлення даних благодійної організації;
- відтворення переліку власних підтверджуючих документів;
- вилучення власного документа;
- визначення нового благодійного проєкту організації;
- оновлення даних благодійного проєкту;
- закриття збору пожертвувань за благодійним проєктом;
- жертвування коштів на благодійний проєкт;
- жертвування коштів на благодійну організацію;
- придбання квитка на благодійну подію;
- відтворення переліку пожертвувань користувача;
- відтворення переліку пожертвувань за благодійним проєктом;
- відтворення переліку допомоги, отриманої користувачем;
- відтворення переліку власних придбаних квитків;
- визначення нової благодійної події з визначенням доступних квитків;
- оновлення даних благодійної події та доступних квитків;
- вподобання благодійної події та скасування його;
- створення закладки на благодійну подію;
- відтворення переліку закладок;
- визначення нової групи квитків на благодійну подію;
- оновлення даних групи квитків на благодійну подію;
- відтворення переліку груп квитків на благодійну подію;
- визначення нової пропозиції потребуючим;
- оновлення даних пропозиції потребуючим;
- додавання елементів до пропозиції;

- оновлення даних окремих елементів пропозиції потребує;
- створення прохання про допомогу;
- оновлення прохання про допомогу;
- створення коментаря до благодійного проєкту;
- створення коментаря до благодійної події;
- створення коментаря до благодійної організації;
- створення закладки на прохання про допомогу;
- перегляд сторінки благодійної організації;
- перегляд сторінки користувача;
- перегляд сторінки благодійного проєкту;
- перегляд сторінки благодійної події;
- перегляд сторінки прохання про допомогу;
- перегляд сторінки пропозиції потребує;
- пошук благодійних організацій;
- пошук благодійних проєктів;
- пошук благодійних подій;
- пошук прохань про допомогу;
- вподобання пропозиції потребує та скасування його;
- відтворення переліку благодійних проєктів організації;
- відтворення переліку діючих благодійних проєктів у місті та загалом;
- відтворення переліку діючих пропозицій потребує у місті;
- відтворення переліку благодійних подій організації;
- відтворення переліку пропозицій потребує організації;
- відтворення переліку прохань про допомогу користувача;
- пошук пропозицій потребує;
- прийняття запиту з пропозицією допомоги благодійному проєкту або його відхилення;
- створення закладки на пропозицію потребує;
- відтворення переліку вподобань;

- створення запиту на пропозицію потребуючим;
- відтворення переліку власних запитів на пропозиції потребуючим;
- прийняття або відхилення запиту на пропозицію потребуючим;
- відтворення переліку запитів на пропозицію потребуючим;
- подання скарги на благодійну організацію;
- відтворення переліку скарг за благодійною організацією;
- відтворення переліку скарг за користувачем;
- подання скарги на користувача;
- відтворення власних створених скарг;
- подання запиту з пропозицією допомоги благодійному проєкту;
- подання запиту з пропозицією допомоги благодійній події;
- подання запиту з пропозицією допомоги благодійній організації;
- відтворення переліку власних запитів з пропозиціями допомоги;
- відтворення переліку запитів з пропозиціями допомоги за благодійним проєктом;
- відтворення переліку запитів з пропозиціями допомоги за благодійною подією;
- відтворення переліку запитів з пропозиціями допомоги за благодійною організацією;
- вподобання благодійного проєкту та скасування його;
- створення закладки на благодійний проєкт;
- подання відповіді на прохання про допомогу;
- відтворення переліку власних відповідей на прохання про допомогу;
- відтворення отриманих відповідей на прохання про допомогу;
- прийняття отриманої відповіді на прохання про допомогу або її відхилення;
- прийняття запиту з пропозицією допомоги благодійній події або його відхилення;
- прийняття запиту з пропозицією допомоги благодійній організації або його відхилення.

A.3.2 Вимоги до складу та параметрів технічних засобів

Апаратні вимоги до серверної сторони:

- процесор у мінімальній конфігурації Intel Core 2 або відповідний еквівалент;
- оперативна пам'ять з доступом не менше ніж до 4 Гб;
- жорсткий диск з бажаним доступним обсягом не менше ніж 100 Гб.

A.4 Порядок контролю та приймання

Основою щодо реалізації контролю та приймання дипломної кваліфікаційної роботи є завдання на роботу і безпосередньо дане технічне завдання. За успішного виконання цих завдань у встановлені строки робота після проходження всіх підготовчих процедур вноситься на захист перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from .models import CharityGroup, Charityproject, City, Offer,
Charityevent, Applicant, ApplicantAsk, ApplicantDocument, Donation,
CharityprojectDonation, CharityGroupDonation, CharityeventTicketDonation,
Offerrequest, CharityprojectSave, ApplicantAskSave, CharityeventSave,
OfferSave, CharityeventTicket, CharityeventLike, CharityprojectLike, OfferLike,
Offereditem, Complaint, CharityprojectVolunteer, CharityeventVolunteer,
CharitygroupVolunteer, ApplicantAskAnswer, Sphere

from datetime import datetime

from django.shortcuts import render

from django.db.models import Q, F

def charitygroup_projects(request, charitygroupSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    charity_group = CharityGroup.objects.get(slug = charitygroupSlug)
    active_now = datetime.now()
    charity_projects = Charityproject.objects.filter(charitygroup =
charity_group, active__lte = active_now).order_by("-saved")
    for charity_project in charity_projects:
        likes = CharityprojectLike.objects.filter(charityproject =
charity_project).count()
        charity_project.likes = likes
        like = CharityprojectLike.objects.filter(charityproject =
charity_project, applicant = applicant)
        if like:
            charity_project.like = True
        else:
            charity_project.like = False
        save = CharityprojectSave.objects.filter(charityproject =
charity_project, applicant = applicant)

```

```

    if save:
        charity_project.save = True
    else:
        charity_project.save= False
    results["projects"] = charity_projects
    return render(request, "charityapp/projects.html", results)

def city_projects(request, cityId = None):
    results = []
    if cityId:
        city = City.objects.get(id = cityId)
        applicant = Applicant.objects.get(id = request.user.id)
        active_now = datetime.now()
        if cityId:
            charity_projects = Charityproject.objects.filter(active__lte =
active_now, cities = city, donated__lte = F("goalsum")).order_by("-saved")
        else:
            charity_projects = Charityproject.objects.filter(active__lte =
active_now, donated__lte = F("goalsum")).order_by("-saved")
        for charity_project in charity_projects:
            likes = CharityprojectLike.objects.filter(charityproject =
charity_project).count()
            charity_project.likes = likes
            like = CharityprojectLike.objects.filter(charityproject =
charity_project, applicant = applicant)
            if like:
                charity_project.like = True
            else:
                charity_project.like = False

```

```

        save = CharityprojectSave.objects.filter(charityproject =
charity_project, applicant = applicant)
        if save:
            charity_project.save = True
        else:
            charity_project.save = False
    results["projects"] = charity_projects
    return render(request, "charityapp/cityprojects.html", results)

def city_offers(request, cityId):
    results = []
    city = City.objects.get(id = cityId)
    applicant = Applicant.objects.get(id = request.user.id)
    charity_offers = Offer.objects.filter(active = True, city =
city).order_by("-saved")
    for charity_offer in charity_offers:
        likes = OfferLike.objects.filter(offer = charity_offer).count()
        like = OfferLike.objects.filter(offer = charity_offer, applicant =
applicant)
        charity_offer.likes = likes
        if like:
            charity_offer.like= True
        else:
            charity_offer.like = False
        save = OfferSave.objects.filter(offer= charity_offer, applicant =
applicant)
        if save:
            charity_offer.save = True
        else:
            charity_offer.save = False

```

```

results["offers"] = charity_offers
return render(request, "charityapp/cityoffers.html", results)

```

```

def charitygroup_events(request, charitygroupSlug):
    results = []
    charity_group = CharityGroup.objects.get(slug = charitygroupSlug)
    applicant = Applicant.objects.get(id = request.user.id)
    active_now = datetime.now()
    charity_events = Charityevent.objects.filter(charitygroup =
charity_group, active__lte = active_now).order_by("-saved")
    for charity_event in charity_events:
        likes = CharityeventLike.objects.filter(charityevent =
charity_event).count()
        charity_event.likes = likes
        like = CharityeventLike.objects.filter(charityevent = charity_event,
applicant = applicant)
        if like:
            charity_event.like = True
        else:
            charity_event.like = False
        save = CharityeventSave.objects.filter(charityevent = charity_event,
applicant = applicant)
        if save:
            charity_event.save = True
        else:
            charity_event.save = False
    results["events"] = charity_events
    return render(request, "charityapp/events.html", results)

```

```

def charitygroup_offers(request, charitygroupSlug):

```

```

results = []
applicant = Applicant.objects.get(id = request.user.id)
charity_group = CharityGroup.objects.get(slug = charitygroupSlug)
charity_offers = Offer.objects.filter(charitygroup =
charity_group).order_by("-saved")
for charity_offer in charity_offers:
    likes = OfferLike.objects.filter(offer = charity_offer).count()
    charity_offer.likes = likes
    like = OfferLike.objects.filter(offer = charity_offer, applicant =
applicant)
    if like:
        charity_offer.like = True
    else:
        charity_offer.like = False
    save = OfferSave.objects.filter(offer= charity_offer, applicant =
applicant)
    if save:
        charity_offer.save = True
    else:
        charity_offer.save = False
results["offers"] = charity_offers
return render(request, "charityapp/offers.html", results)

```

```

def applicant_asks(request, applicantLogin):
    results = []
    applicant = Applicant.objects.get(login = applicantLogin)
    my_applicant = Applicant.objects.get(id = request.user.id)
    applicant_asks = ApplicantAsk.objects.filter(applicant =
applicant).order_by("-saved")
    for applicant_ask in applicant_asks:

```

```

        save = ApplicantAskSave.objects.filter(applicantask = applicant_ask,
applicant = my_applicant)
        if save:
            applicant_ask.save = True
        else:
            applicant_ask.save = False
    results["asks"] = applicant_asks
    return render(request, "charityapp/asks.html", results)

```

```

def applicant_documents(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    applicant_documents = ApplicantDocument.objects.filter(applicant =
applicant).order_by("-id")
    results["documents"] = applicant_documents
    return render(request, "charityapp/documents.html", results)

```

```

def applicant_donations(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    donations = Donation.objects.filter(applicant = applicant).order_by("-
saved")
    for donation in donations:
        charityproject =
CharityprojectDonation.objects.prefetch_related("charityproject").filter(donation =
donation).get().value("charityproject")
        if charityproject:
            donation.goal = charityproject
        else:

```

```

        charitygroup =
CharityGroupDonation.objects.prefetch_related("charitygroup").filter(donation =
donation).get().value("charitygroup")
        if charitygroup:
            donation.goal = charitygroup
        else:
            charityeventticket =
CharityeventTicketDonation.objects.prefetch_related("charityeventticket").filter(d
onation = donation).get().value("charityeventticket")
            if charityeventticket:
                donation.goal = charityeventticket
        results["donations"] = donations
        return render(request, "charityapp/donations.html", results)

def charityproject_donations(request, charityprojectSlug):
    results = []
    charity_project = Charityproject.objects.get(slug = charityprojectSlug)
    donations =
CharityprojectDonation.objects.prefetch_related("donation").filter(charityproject =
charity_project, donation__paid = True).order_by("-donation__saved")
    results["donations"] = donations
    return render(request, "charityapp/projectdonations.html", results)

def account_aids(request, accountLogin = None, mode = False):
    results = []
    if accountLogin:
        applicant = Applicant.objects.get(login = accountLogin)
    else:
        applicant = Applicant.objects.get(id = request.user.id)
    if not mode:

```

```

        offer_requests =
Offerrequest.objects.prefetch_related("offereditem").prefetch_related("offer__offer
editem").filter(applicant = applicant).order_by("-saved")
        asks = ApplicantAsk.objects.filter(applicant = applicant).order_by("-
saved")
    else:
        offer_requests =
Offerrequest.objects.prefetch_related("offereditem").prefetch_related("offer__offer
editem").filter(applicant = applicant, active = 1).order_by("-saved")
        asks = ApplicantAsk.objects.filter(applicant = applicant, active =
1).order_by("-saved")
        results["offers"] = offer_requests
        results["asks"] = asks
        return render(request, "charityapp/aids.html", results)

```

```

def applicant_tickets(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    donations = Donation.objects.filter(applicant = applicant, paid =
True).order_by("-saved")
    tickets =
CharityeventTicketDonation.objects.prefetch_related("charityeventticket").prefetc
h_related("charityeventticket__charityevent").filter(donation__in = donations)
    results["tickets"] = tickets
    return render(request, "charityapp/tickets.html", results)

```

```

def applicant_saves(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)

```

```

        charityprojects =
CharityprojectSave.objects.prefetch_related("charityproject").prefetch_related("ch
arityproject__charitygroup").filter(applicant = applicant).order_by("-id")
        for charity_project in charityprojects:
            likes = CharityprojectLike.objects.filter(charityproject =
charity_project.charityproject).count()
            charity_project.likes = likes
            like = CharityprojectLike.objects.filter(charityproject =
charity_project.charityproject, applicant = applicant)
            if like:
                charity_project.like = True
            else:
                charity_project.like = False
        results["projects"] = charityprojects
        asks =
ApplicantAskSave.objects.prefetch_related("applicantask").prefetch_related("appli
cant").filter(applicant = applicant).order_by("-id")
        for applicant_ask in asks:
            save = ApplicantAskSave.objects.filter(applicantask =
applicant_ask.applicantask, applicant = applicant)
            if save:
                applicant_ask.save = True
            else:
                applicant_ask.save = False
        results["asks"] = asks
        charityevents =
CharityeventSave.objects.prefetch_related("charityevent").prefetch_related("charit
yevent__charitygroup").filter(applicant = applicant).order_by("-id")
        for charity_event in charityevents:

```

```

        likes = CharityeventLike.objects.filter(charityevent =
charity_event.charityevent).count()
        charity_event.likes = likes
        like = CharityeventLike.objects.filter(charityevent =
charity_event.charityevent, applicant = applicant)
        if like:
            charity_event.like = True
        else:
            charity_event.like = False
    results["events"] = charityevents
    offers =
OfferSave.objects.prefetch_related("offer").prefetch_related("offer__charitygroup"
).filter(applicant = applicant).order_by("-id")
    for charity_offer in offers:
        likes = OfferLike.objects.filter(offer = charity_offer.offer).count()
        charity_offer.likes = likes
        like = OfferLike.objects.filter(offer = charity_offer.offer, applicant =
applicant)
        if like:
            charity_offer.like = True
        else:
            charity_offer.like = False
    results["offers"] = offers
    return render(request, "charityapp/saves.html", results)

def event_tickets(request, eventSlug):
    results = []
    event = Charityevent.objects.get(slug = eventSlug)
    tickets = CharityeventTicket.objects.filter(charityevent =
event).order_by("id")

```

```

results["tickets"] = tickets
return render(request, "charityapp/eventtickets.html", results)

def applicant_likes(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    charityprojects =
CharityprojectLike.objects.prefetch_related("charityproject").prefetch_related("cha
rityproject__charitygroup").filter(applicant = applicant).order_by("-id")
    for charityproject in charityprojects:
        save = CharityprojectSave.objects.filter(charityproject =
charityproject.charityproject, applicant = applicant)
        if save:
            charityproject.save = True
        else:
            charityproject.save = False
    results["projects"] = charityprojects
    charityevents =
CharityeventLike.objects.prefetch_related("charityevent").prefetch_related("charit
yevent__charitygroup").filter(applicant = applicant).order_by("-id")
    for charityevent in charityevents:
        save = CharityeventSave.objects.filter(charityevent =
charityevent.charityevent, applicant = applicant)
        if save:
            charityevent.save = True
        else:
            charityevent.save = False
    results["events"] = charityevents

```

```

        offers =
OfferLike.objects.prefetch_related("offer").prefetch_related("offer__charitygroup"
).filter(applicant = applicant).order_by("-id")
        for offer in offers:
            save = OfferSave.objects.filter(offer = offer.offer, applicant =
applicant)
            if save:
                offer.save = True
            else:
                offer.save = False
        results["offers"] = offers
        return render(request, "charityapp/likes.html", results)

def applicant_requests(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    requests =
Offerrequest.objects.select_related("offereditem").select_related("offereditem__ite
msubcategory").select_related("offereditem__offer").filter(applicant =
applicant).order_by("-saved")
    results["requests"] = requests
    return render(request, "charityapp/requests.html", results)

def offer_requests(request, offerSlug = None, offereditemId = None):
    results = []
    if offer:
        offer = Offer.objects.select_related("charitygroup").filter(slug =
offerSlug).get()
    if offereditemId:

```

```

        offereditem =
Offereditem.objects.select_related("offer").select_related("itemsubcategory").filter
(pk = offereditemId).get()
        applicant = Applicant.objects.prefetch_related("charitygroups").(id =
request.user.id).get()
        if applicant.role == True and offer.charitygroup in
applicant.charitygroups:
            if offer:
                requests =
Offerrequest.objects.select_related("offereditem").select_related("offereditem__ite
msubcategory").select_related("offereditem__offer").filter(offereditem__offer =
offer).order_by("-saved")
            elif offereditemId:
                requests =
Offerrequest.objects.select_related("offereditem__itemsubcategory").select_related
("offereditem__offer").filter(offereditem = offereditem).order_by("-saved")
            results["requests"] = requests
        return render(request, "charityapp/offerrequests.html", results)

def charitygroup_complaints(request, charitygroupSlug = None):
    results = []
    applicant = Applicant.objects.prefetch_related("charitygroups").get(id =
request.user.id)
    if charitygroupSlug:
        charitygroup = CharityGroup.objects.filter(slug =
charitygroupSlug).get()
    else:
        charitygroups = applicant.charitygroup
    if charitygroupSlug:

```

```

        complaints =
Complaint.objects.select_related("applicant").filter(charitygroup = charitygroup,
standard = True).order_by("-saved")
    else:
        complaints =
Complaint.objects.select_related("applicant").filter(charitygroup__in =
charitygroups, standard = True).order_by("-saved")
    results["complaints"] = complaints
    return render(request, "charityapp/groupcomplaints.html", results)

```

```

def applicant_complaints(request, applicantLogin = None):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    if applicantLogin:
        applicant = Applicant.objects.get(login = applicantLogin)
    complaints =
Complaint.objects.select_related("charitygroup").filter(applicant = applicant,
standard = False).order_by("-saved")
    results["complaints"] = complaints
    return render(request, "charityapp/appcomplaints.html", results)

```

```

def own_applicant_complaints(request):
    results = []
    applicant = Applicant.objects.prefetch_related("charitygroups").filter(id
= request.user.id).get()
    if applicant.role:
        complaints =
Complaint.objects.select_related("charitygroup").filter(charitygroup__in =
applicant.charitygroups, standard = False).order_by("-saved")
    else:

```

```

        complaints =
Complaint.objects.select_related("charitygroup").filter(applicant = applicant,
standard = True).order_by("-saved")
        results["complaints"] = complaints
        return render(request, "charityapp/owncomplaints.html", results)

def own_volunteers(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    charityprojects =
CharityprojectVolunteer.objects.select_related("charityproject").select_related("vol
unteer").select_related("charityproject__charitygroup").filter(volunteer__applicant
= applicant)
    results["projects"] = charityprojects
    charityevents =
CharityeventVolunteer.objects.select_related("charityevent").select_related("volun
teer").select_related("charityevent__charitygroup").filter(volunteer__applicant =
applicant)
    results["events"] = charityevents
    charitygroups =
CharitygroupVolunteer.objects.select_related("charitergroup").select_related("volu
nteer").filter(volunteer__applicant = applicant)
    results["charitygroups"] = charitygroups
    return render(request, "charityapp/volunteers.html", results)

def project_volunteers(request, projectSlug):
    results = []
    charityproject = Charityproject.objects.get(slug = projectSlug)

```

```

        volunteers =
CharityprojectVolunteer.objects.select_related("charityproject").select_related("vol
unteer").filter(charityproject = charityproject).values("volunteer")
        results["volunteers"] = volunteers
        return render(request, "charityapp/projectvolunteers.html", results)

def event_volunteers(request, charityeventSlug):
    results = []
    charityevent = Charityevent.objects.get(slug = charityeventSlug)
    volunteers =
CharityeventVolunteer.objects.select_related("charityevent").select_related("volun
teer").filter(charityevent = charityevent).values("volunteer")
    results["volunteers"] = volunteers
    return render(request, "charityapp/eventvolunteers.html", results)

def group_volunteers(request, charitygroupSlug):
    results = []
    charitygroup = CharityGroup.objects.get(slug = charitygroupSlug)
    volunteers =
CharitygroupVolunteer.objects.select_related("charitergroup").select_related("volu
nteer").filter(charitergroup = charitygroup).values("volunteer")
    results["volunteers"] = volunteers
    return render(request, "charityapp/groupvolunteers.html", results)

def applicant_ask_answers(request, applicantaskSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    ask = ApplicantAsk.objects.select_related("applicant").filter(slug =
applicantaskSlug).get()
    if ask.applicant == applicant:

```

```

        answers = ApplicantAskAnswer.objects.filter(applicantask =
ask).order_by("-saved")
        results["answers"] = answers
        return render(request, "charityapp/askanswers.html", results)

```

```

def own_ask_answers(request):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    answers =
ApplicantAskAnswer.objects.select_related("applicantask").select_related("applica
ntask__sphere").select_related("applicantask__applicant").filter(applicant =
applicant).order_by("-saved")
    results["answers"] = answers
    return render(request, "charityapp/ownanswers.html", results)

```

```

def charity_group(request, charitygroupSlug):
    results = []
    charitygroup = CharityGroup.objects.get(slug = charitygroupSlug)
    charityprojects =
Charityproject.objects.select_related("sphere").prefetch_related("cities").filter(char
itygroup = charitygroup)[:3]
    applicant = Applicant.objects.get(id = request.user.id)
    for charity_project in charityprojects:
        likes = CharityprojectLike.objects.filter(charityproject =
charity_project).count()
        charity_project.likes = likes
        like = CharityprojectLike.objects.filter(charityproject =
charity_project, applicant = applicant)
        if like:
            charity_project.like = True

```

```

else:
    charity_project.like = False
    save = CharityprojectSave.objects.filter(charityproject =
charity_project, applicant = applicant)
    if save:
        charity_project.save = True
    else:
        charity_project.save= False
results["projects"] = charityprojects
results["group"] = charitygroup
charityevents =
Charityevent.objects.select_related("city").filter(charitygroup = charitygroup)[:3]
for charity_event in charityevents:
    likes = CharityeventLike.objects.filter(charityevent =
charity_event).count()
    charity_event.likes = likes
    like = CharityeventLike.objects.filter(charityevent = charity_event,
applicant = applicant)
    if like:
        charity_event.like = True
    else:
        charity_event.like = False
    save = CharityeventSave.objects.filter(charityevent = charity_event,
applicant = applicant)
    if save:
        charity_event.save = True
    else:
        charity_event.save = False
results["events"] = charityevents

```

```

offers = Offer.objects.select_related("city").filter(charitygroup =
charitygroup)[:3]
for charity_offer in offers:
    likes = OfferLike.objects.filter(offer = charity_offer).count()
    charity_offer.likes = likes
    like = OfferLike.objects.filter(offer = charity_offer, applicant =
applicant)
    if like:
        charity_offer.like = True
    else:
        charity_offer.like = False
    save = OfferSave.objects.filter(offer= charity_offer, applicant =
applicant)
    if save:
        charity_offer.save = True
    else:
        charity_offer.save = False
results["offers"] = offers
return render(request, "charityapp/charitygroup.html", results)

```

```

def charity_project(request, charityprojectSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    charityproject =
Charityproject.objects.select_related("sphere").prefetch_related("cities").filter(slug
= charityprojectSlug).get()
    positions = CharityprojectPosition.objects.filter(charityproject =
charityproject)

```

```

        comments =
CharityprojectComment.objects.select_related("comment").filter(charityproject =
charityproject).order_by("-comment__saved")
        likes = CharityprojectLike.objects.filter(charityproject =
charityproject).count()
        charityproject.likes = likes
        like = CharityprojectLike.objects.filter(charityproject = charityproject,
applicant = applicant)
        if like:
            charityproject.like = True
        else:
            charityproject.like = False
        save = CharityprojectSave.objects.filter(charityproject = charityproject,
applicant = applicant)
        if save:
            charityproject.save = True
        else:
            charityproject.save= False
        results["project"] = charityproject
        results["positions"] = positions
        results["comments"] = comments
        return render(request, "charityapp/charityproject.html", results)

def applicant(request, applicantLogin = None):
    results = []
    applicant = Applicant.objects.get(login = applicantLogin)
    if not applicantLogin:
        applicant = Applicant.objects.get(id = request.user.id)
    asks = ApplicantAsk.objects.select_related("sphere").filter(applicant =
applicant).order_by("-saved")

```

```

    for applicant_ask in asks:
        save = ApplicantAskSave.objects.filter(applicantask = applicant_ask,
applicant = applicant)
        if save:
            applicant_ask.save = True
        else:
            applicant_ask.save = False
    results["applicant"] = applicant
    results["asks"] = asks
    return render(request, "charityapp/applicant.html", results)

```

```

def charity_event(request, charityeventSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    charityevent =
Charityevent.objects.select_related("charitygroup").select_related("city").filter(slu
g = charityeventSlug).get()
    tickets = CharityeventTicket.objects.filter(charityevent = charityevent)
    comments =
CharityeventComment.objects.select_related("comment").filter(charityevent =
charityevent).order_by("-comment__saved")
    likes = CharityeventLike.objects.filter(charityevent =
charityevent).count()
    charityevent.likes = likes
    like = CharityeventLike.objects.filter(charityevent = charityevent,
applicant = applicant)
    if like:
        charityevent.like = True
    else:
        charityevent.like = False

```

```

        save = CharityeventSave.objects.filter(charityevent = charityevent,
applicant = applicant)
        if save:
            charityevent.save = True
        else:
            charityevent.save = False
        results["event"] = charityevent
        results["tickets"] = tickets
        results["comments"] = comments
        return render(request, "charityapp/charityevent.html", results)

```

```

def applicant_ask(request, applicantaskSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)
    applicantask =
ApplicantAsk.objects.select_related("sphere").select_related("applicant").filter(slu
g = applicantaskSlug).get()
        save = ApplicantAskSave.objects.filter(applicantask = applicantask,
applicant = applicant)
        if save:
            applicantask.save = True
        else:
            applicantask.save = False
        results["ask"] = applicantask
        return render(request, "charityapp/applicantask.html", results)

```

```

def offer(request, offerSlug):
    results = []
    applicant = Applicant.objects.get(id = request.user.id)

```

```

        offer =
Offer.objects.select_related("charitygroup").select_related("city").filter(slug =
offerSlug).get()
        likes = OfferLike.objects.filter(offer = offer).count()
        charity_offer.likes = likes
        like = OfferLike.objects.filter(offer = offer, applicant = applicant)
        if like:
            offer.like = True
        else:
            offer.like = False
        save = OfferSave.objects.filter(offer = offer, applicant = applicant)
        if save:
            offer.save = True
        else:
            offer.save = False
        results["offer"] = offer
        return render(request, "charityapp/offer.html", results)

```

```

def charity_group_find(request, charityTitle = None, cityName = None):
    results = []
    done = False
    if charityTitle:
        charities = CharityGroup.objects.filter(title__icontains = charityTitle)
        done = True
    if done and cityName:
        charities = charities.filter(city__name = cityName)
    elif not done and cityName:
        charities = CharityGroup.objects.filter(city__name = cityName)
    results["charities"] = charities
    results["cities"] = City.objects.all()

```

```
return render(request, "charityapp/findcharities.html", results)
```

```
def charity_project_find(request, sphereTitle = None, projectTitle = None,
minSum = None, maxSum = None, donatedSum = None, active = 0, periodStart =
None, periodEnd = None, cityName = None, minLikes = None, minComments =
None):
    results = []
    active_now = datetime.now()
    charityprojects = Charityproject.objects.filter(active__lte = active_now)
    if sphereTitle:
        charityprojects = charityprojects.filter(sphere__title = sphereTitle)
    if projectTitle:
        charityprojects = charityprojects.filter(title__icontains = projectTitle)
    if minSum:
        charityprojects = charityprojects.filter(goalsum__gte = minSum)
    if maxSum:
        charityprojects = charityprojects.filter(goalsum__lte = maxSum)
    if donatedSum:
        charityprojects = charityprojects.filter(donated__gte = donatedSum)
    if periodStart:
        charityprojects = charityprojects.filter(active__gte = periodStart)
    if periodEnd:
        charityprojects = charityprojects.filter(active__lte = periodEnd)
    if cityName:
        city = City.objects.get(name = cityName)
        charityprojects = charityprojects.filter(cities = city)
    for charity_project in charityprojects:
        likes = CharityprojectLike.objects.filter(charityproject =
charity_project).count()
        if not minLikes or (minLikes and likes >= minLikes):
```

```

        comments =
CharityprojectComment.objects.select_related("comment").filter(charityproject=
charity_project).count()

        if not minComments or (minComments and comments >=
minComments)

            if (active == 0 and not charity_project.ended) or (active == 0 and
charity_project.ended and charity_project.ended >= periodStart and
charity_project.ended <= periodEnd) or (active == 1 and not charity_project.ended
and charity_project.goalsum > charity_project.donated) or (active == 2 and
charity_project.ended and charity_project.ended >= periodStart and
charity_project.ended <= periodEnd and charity_project.goalsum <=
charity_project.donated) or (active == 3 and charity_project.ended and
charity_project.ended >= periodStart and charity_project.ended <= periodEnd and
charity_project.goalsum > charity_project.donated):

                charity_project.likes = likes

                like = CharityprojectLike.objects.filter(charityproject =
charity_project, applicant = applicant)

                if like:

                    charity_project.like = True

                else:

                    charity_project.like = False

                save = CharityprojectSave.objects.filter(charityproject =
charity_project, applicant = applicant)

                if save:

                    charity_project.save = True

                else:

                    charity_project.save= False

    results["projects"] = charityprojects
    results["cities"] = City.objects.all()
    results["spheres"] = Sphere.objects.all()

```

```
return render(request, "charityapp/findprojects.html", results)
```

```
def applicant_ask_find(request, sphereTitle = None, askTitle = None,
cityName = None):
    results = []
    asks = ApplicantAsk.objects.filter(active = True)
    if sphereTitle:
        asks = asks.filter(sphere__title = sphereTitle)
    if askTitle:
        asks = asks.filter(title__icontains = askTitle)
    if cityName:
        asks = asks.filter(city__name = cityName)
    applicant = Applicant.objects.get(id = request.user.id)
    for ask in asks:
        save = ApplicantAskSave.objects.filter(applicantask = ask, applicant =
applicant)
        if save:
            ask.save = True
        else:
            ask.save = False
    results["asks"] = asks
    results["cities"] = City.objects.all()
    results["spheres"] = Sphere.objects.all()
    return render(request, "charityapp/findask.html", results)
```

```
def charity_events_find(request, charitygroupSlug = None, eventTitle =
None, cityName = None, eventStart = None, eventEnd = None, minSum = None,
maxSum = None, donatedSum = None, active = 0, minLikes = None,
minComments = None):
    results = []
```

```

active_now = datetime.now()
charityevents = Charityevent.objects.filter(active__lte = active_now)
if charitygroupSlug:
    charityevents = charityevents.filter(charitygroup = charitygroupSlug)
if eventTitle:
    charityevents = charityevents.filter(title__icontains = eventTitle)
if minSum:
    charityevents = charityevents.filter(goalsum__gte = minSum)
if maxSum:
    charityevents = charityevents.filter(goalsum__lte = maxSum)
if donatedSum:
    charityevents = charityevents.filter(donated__gte = donatedSum)
if eventStart:
    charityevents = charityevents.filter(eventstart__gte = eventStart)
if periodEnd:
    charityevents = charityevents.filter(eventstart__lte = eventEnd)
if cityName:
    charityevents = charityevents.filter(city__name = cityName)
for charity_event in charityevents:
    likes = CharityeventLike.objects.filter(charityevent =
charity_event).count()
    if not minLikes or (minLikes and likes >= minLikes):
        comments =
CharityeventComment.objects.select_related("comment").filter(charityevent =
charity_event).count()
        if not minComments or (minComments and comments >=
minComments)
            if (active == 0) or (active == 1 and charity_event.goalsum >
charity_event.donated and charity_event.eventstart > active_now) or (active == 2
and charity_event.goalsum <= charity_event.donated) or (active == 3 and

```

```
charity_event.goalsum > charity_event.donated and charity_event.eventstart <
active_now):
```

```
    charity_event.likes = likes
```

```
    like = CharityeventLike.objects.filter(charityevent =
charity_event, applicant = applicant)
```

```
    if like:
```

```
        charity_event.like = True
```

```
    else:
```

```
        charity_event.like = False
```

```
    save = CharityeventSave.objects.filter(charityevent =
charity_event, applicant = applicant)
```

```
    if save:
```

```
        charity_event.save = True
```

```
    else:
```

```
        charity_event.save= False
```

```
results["events"] = charityevents
```

```
results["cities"] = City.objects.all()
```

```
results["charities"] = CharityGroup.objects.all()
```

```
return render(request, "charityapp/findevents.html", results)
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра на тему:

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ БЛАГОДІЙНОЇ ДОПОМОГИ SOFTWARE FOR CHARITY SUPPORT

Виконав студент групи КНТ-219
Керівник доцент, к.т.н.

Даніїл ЛАЗУТІН
Степан СКРУПСЬКИЙ

2023

Рисунок В.1 – Слайд титульний

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- об'єкт роботи – процес надання благодійної допомоги;
- предмет роботи – програмне забезпечення для благодійної допомоги;
- мета роботи – розробити програмне забезпечення для благодійної допомоги.

2

Рисунок В.2 – Об'єкт, предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- аналіз програмних аналогів і визначення концепції програми;
- проектування програмного забезпечення для благодійної допомоги;
- реалізація вебдодатку для благодійної допомоги.

3

Рисунок В.3 – Завдання роботи

АНАЛІЗ ПРОГРАМНИХ АНАЛОГІВ

Ознака	Фонд Сергія Притули	Hivebrite	Дипломна робота
Інформування про благодійні проекти організації	+	+	+
Збір коштів на фінансування проектів	+	+	+
Підтримка керування продажем квитків на благодійні події	–	+	+
Запитування допомоги самими користувачами	+	–	+
Інформування про наявні благодійні пропозиції	–	Тільки стосовно роботи	+
Надання допомоги користувачам іншими користувачами	–	–	+

4

Рисунок В.4 – Аналіз програмних аналогів

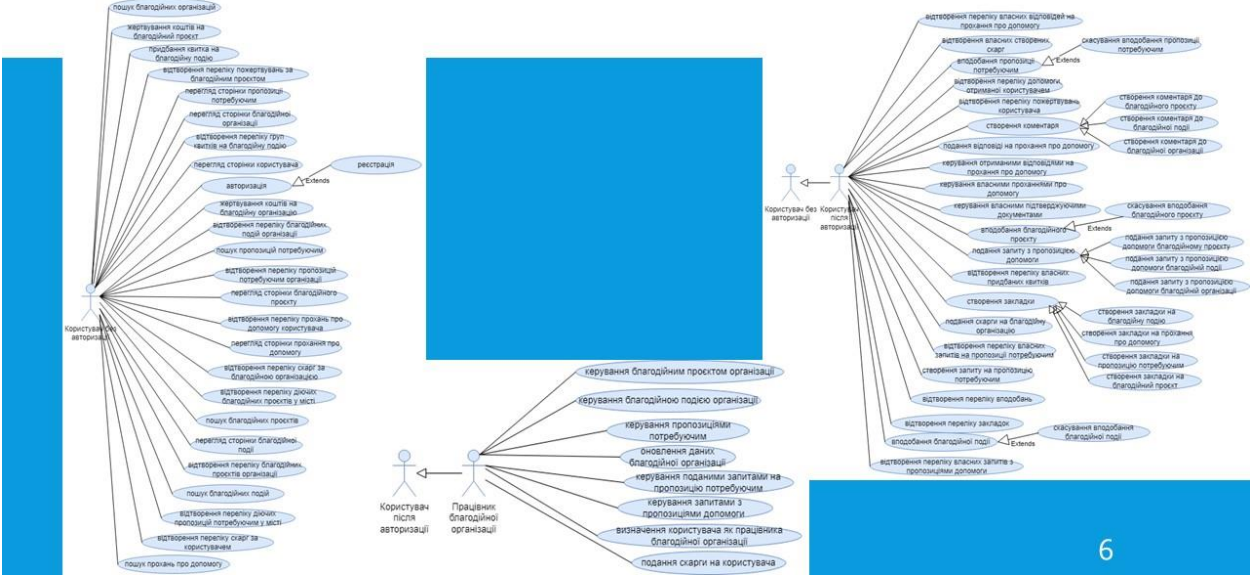
ПОРІВНЯННЯ ФРЕЙМВОРКІВ ВЕБРОЗРОБКИ

Ознака	Django	Spring
Мова програмування	Python	Java
Архітектура	Model-view-template	Model-view-controller
Складність вивчення	Низька	Висока
Швидкість роботи	Середня	Висока
Пошук даних	Доступний в тому числі повнотекстовий	Повнотекстовий пошук можливий тільки з зовнішніми технологіями
Наявні засоби	Має багато вбудованих засобів, можуть встановлюватися додаткові модулі	В основному потребує встановлення додаткових модулів

5

Рисунок В.5 – Порівняння фреймворків веброзробки

ДІАГРАМИ ПРЕЦЕДЕНТІВ



6

Рисунок В.6 – Діаграми прецедентів

[illegible]

ПОШУК БЛАГОДІЙНИХ ПРОЄКТІВ

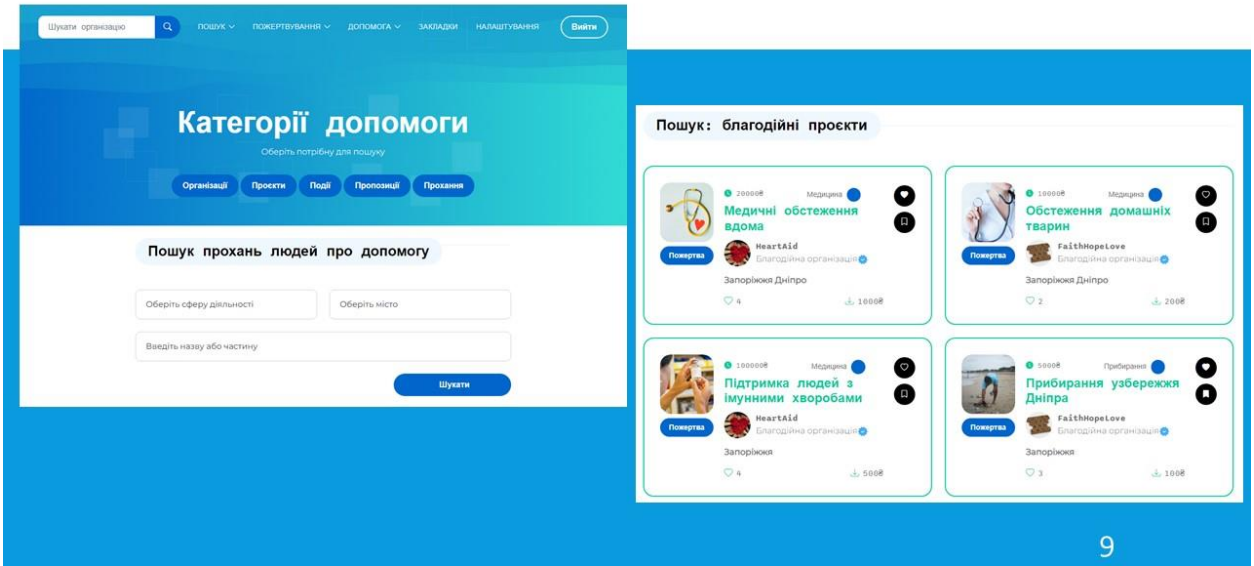


Рисунок В.9 – Пошук благодійних проєктів

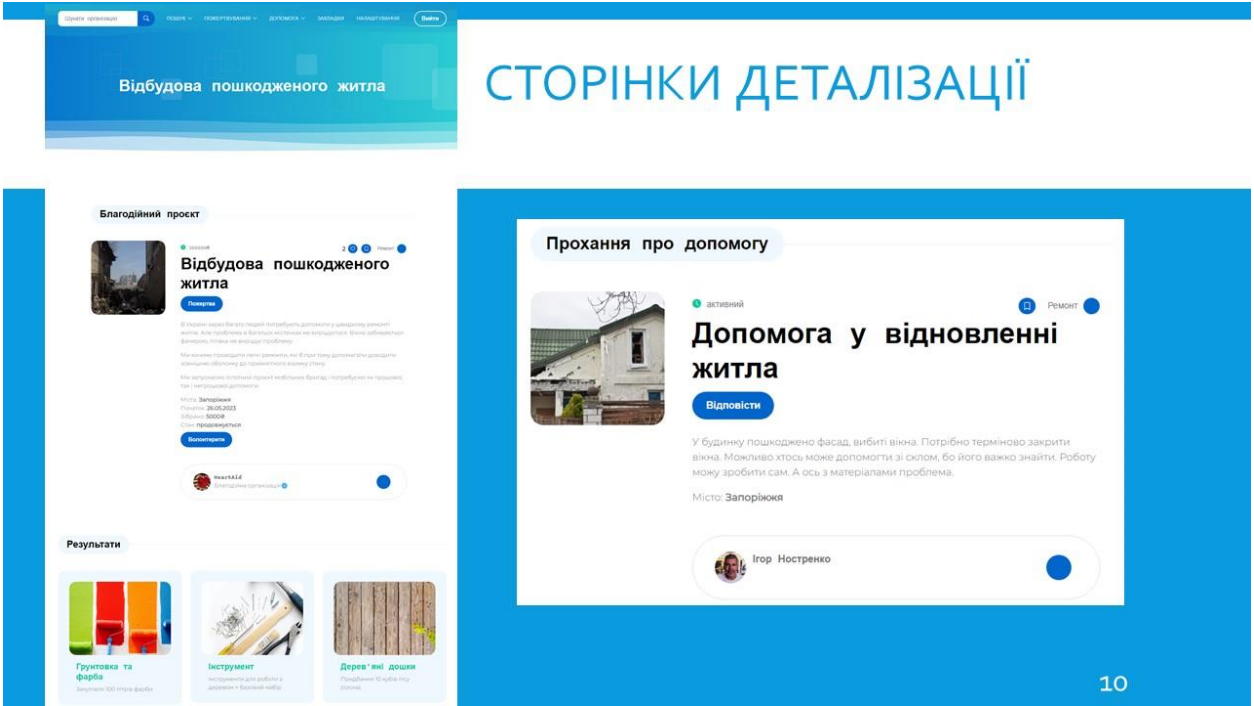


Рисунок В.10 – Сторінки деталізації

ПРОПОЗИЦІЇ ПОТРЕБУЮЧИМ БЛАГОДІЙНОЇ ОРГАНІЗАЦІЇ

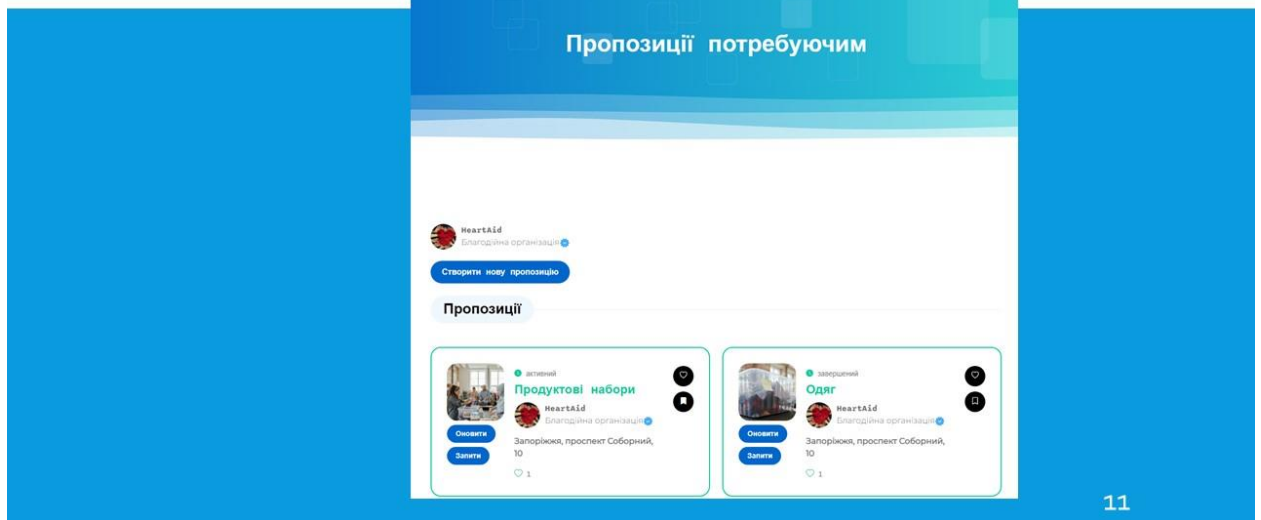


Рисунок В.11 – Пропозиції потребуєчим благодійної організації

ВИСНОВКИ

- для забезпечення взаємодії благодійних організацій з людьми та між людьми напряду створено програмне забезпечення в цій дипломній кваліфікаційний роботі;
- набір проєктних рішень щодо даного програмного забезпечення включив: мову програмування Python і систему керування базами даних PostgreSQL як засоби проєктування і розробки, моделювання вимог до програми через діаграми прецедентів для користувачів-кандидатів, які претендують на благодійну допомогу, та для працівників благодійних організацій, результати проєктування бази даних;
- розроблена програма призначена для підтримки роботи благодійників та благодійних організацій в частині отримання ресурсу для допомоги та знаходження потребуєчим, щоби визначити, чи варто надавати їм допомогу, а також кому саме конкретну допомогу надавати: у повній мірі заплановані функції реалізовані у складі вебсистеми з підтримкою фреймворку Django, визначенням представлень на основі створення функцій, а моделей і форм як класів.

Рисунок В.12 – Висновки