Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями

Виконав: студент IV курсу
групи 2ПІ-20Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Богачук Дмитро Володимирович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Войцеховська О.В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.

«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Богачуку Дмитру Володимировичу

1. Тема роботи – розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями.

Керівник роботи: Чехместрук Роман Юрійович, к.т.н. доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: середовище розробки WebStorm, мова розробки JavaScript, бібліотека React, протокол передачі даних HTTP, системи управління базою даних MongoDB, алгоритм кластеризації.

4. Зміст текстової частини: вступ, обгрунтування вибору методу розробки та постанова задачі дослідження; порівняльний аналіз аналогів; розрозбка графічного інтерфейсу, моделі та алгоритмів веб-застосунку; розробка програмних модулів веб-застосунку; тестування веб-застосунку; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: актуальність; мета, об'єкт та предмет дослідження; задачі дослідження; практичне значення; порівняльний аналіз аналогів; графічний інтерфейс онлайн-порталу; модель роботи онлайн-порталу; блок-схеми алгоритмів роботи онлайн-порталу, тестування онлайн-порталу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Чехместрук Р.Ю., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.2

7. Дата видачі завдання 13 березня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз стану наявних онлайн-порталів для біженців	13.03.24 – 31.03.24	Вик.
2	Розробка інтерфейсу, моделі та алгоритмів програмного продукту	01.04.24 – 20.04.24	Вик.
3	Вибір середовища та мови розробки	21.04.24 – 04.05.24	Вик.
4	Розробка модуля кластеризації маркерів	05.05.24 – 14.05.24	Вик.
5	Розробка модуля для відображення маркерів на інтерактивній карті	15.05.24 – 19.05.24	Вик.
6	Тестування	20.05.24 – 29.05.24	Вик.
7	Оформлення матеріалів до захисту БДР	30.05.24 – 31.05.24	Вик.

Студент _____ **Д. В. Бог**
(підпис) (ініціали та прі

Керівник бакалаврської дипломної роботи _____ **Р. Ю. Чехмест**
(підпис) (ініціали та пр

АНОТАЦІЯ

УДК 004.912.032.26

Богачук Д.В. Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 59 с.

На укр. мові. Бібліогр. : 25 назв ; рис. : 39 ; табл. 3.

У бакалаврській дипломній роботі проведено детальний аналіз методів і засобів розробки інтегровного онлайн-порталу для біженців з персоналізованими рекомендаціями. Запропоновано новий підхід до підтримки біженців, інтегруючи інтерактивну мапу, створену за допомогою React-Leaflet, та модуль новин в один онлайн-портал. Це дозволяє надавати користувачам актуальну інформацію та персоналізовані рекомендації на основі їхнього місцезнаходження.

Розроблено моделі, алгоритми та програмні модулі для відображення маркерів на інтерактивній карті й кластеризації маркерів для полегшення навігації користувачів та зменшення навантаження на застосунок. Програмні модулі написані з використанням мови програмування JavaScript, середовища node.js, фреймворку Express.js, а також бібліотек React, use-superclaster, Leaflet та Mongoose у інтегрованому середовищі розробки WebStorm.

Отримані в бакалаврській кваліфікаційній роботі результати можна використати для побудови інтегрованого олайн-порталу для біженців з персоналізованими рекомендаціями.

Ключові слова: міграція, біженці, соціальна інтеграція, онлайн-портал, інтерактивна карта, геодані.

‘

ABSTRACT

Bogachuk D.V. Development of an integrated online portal for refugees with personalized recommendations. Vinnitsa: VNTU, 2024. 59 p.

In Ukrainian language. Bibliographer: 25 titles ; fig. : 39; tabl. 3.

A detailed analysis of the methods and means of developing an integrated online portal for refugees with personalized recommendations was carried out in the bachelor thesis. A new approach to refugee support is proposed, integrating an interactive map created with React-Leaflet and a news module into one online portal. This allows users to be provided with up-to-date information and personalized recommendations based on their location.

Models, algorithms and software modules have been developed for displaying markers on an interactive map and clustering markers to facilitate user navigation and reduce the load on the application. The software modules are written using the JavaScript programming language, the node.js environment, the Express.js framework, as well as the React, use-supercluster, Leaflet and Mongoose libraries in the WebStorm integrated development environment.

The results obtained in the bachelor qualification work can be used to build an integrated online portal for refugees with personalized recommendations.

Keywords: migration, refugees, social integration, online portal, interactive map, geodata.

‘

ЗМІСТ

ВСТУП

Обгрунтування вибору теми дослідження. У сучасному світі, де глобальна міграція стає все більш поширеною через різноманітні фактори, такі як конфлікти, кризи та економічні умови, важливість ефективних інструментів та платформ для підтримки біженців у процесі адаптації та соціальної інтеграції стає все більш очевидною. Ці інструменти та платформи не лише допомагають біженцям знайти необхідну інформацію та ресурси, але й сприяють створенню спільноти, яка може надати їм підтримку та співчуття.

Актуальність. У 2022 році світ став свідком значного зростання міграції через події, такі як конфлікт в Україні. Ця ситуація призвела до великого потоку біженців, яких оцінки перевищили п’ять мільйонів осіб у 2022 році [1], а у 2023 році ця кількість збільшилася майже до шести мільйонів [1]. Відповідно, виникає необхідність забезпечити швидкий доступ біженців до інформації про гуманітарну допомогу та можливість моніторингу постраждалих територій.

У цьому контексті важливо враховувати соціальну активність та готовність суспільства допомагати. Наприклад, у країнах Європи, таких як Польща, Німеччина та Франція, волонтерська робота відіграє значну роль у підтримці біженців. У першому кварталі 2022 року волонтерську діяльність підтримали понад 8,5 мільйонів осіб [1]. Це свідчить про високий рівень соціальної відповідальності та готовності громадян допомагати тим, хто потребує підтримки.

Доступність - розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями дозволить забезпечити швидкий доступ до необхідної інформації та ресурсів.

Простота - портал буде розроблений таким чином, щоб бути інтуїтивно зрозумілим та простим у використанні для всіх користувачів, незалежно від їхнього технічного рівня.

Інформативність - портал буде надавати корисну та актуальну інформацію, яка допоможе біженцям у процесі адаптації та інтеграції в нове середовище.

Зручність - завдяки інтеграції різних модулів, портал стане цінним інструментом для спільноти біженців, дозволяючи їм отримувати актуальну інформацію та підтримку.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета і завдання дослідження: Метою дослідження є підвищення доступу до інформації про допомогу біженцям. Це буде важливим кроком у забезпеченні гуманітарної допомоги та соціальної інтеграції для тих, хто опинився у складних життєвих обставинах через міграцію.

Головними завданнями, які необхідно вирішити під час реалізації проєкту, є розробка інтерактивної мапи з можливістю перегляду різних пропозицій допомоги створених волонтерами, розділ з новинами та постами, де біженці можуть стежити за актуальними подіями у світі, профілів користувачів з певними ролями, а також розробка власної карти волонтера з історією створеної допомоги, яка дозволить волонтерам додавати, редагувати та видаляти власні пропозиції.

Основними задачами дослідження є:

– розробка модуля для кластеризації маркерів на мапі, що дозволить групувати подібні об'єкти для зручності навігації користувачів;

– розробка алгоритму для збереження історії маркерів, для перегляду попередніх запитів та дій на мапі;

– розробка власної карти волонтера для можливості створення нової допомоги;

– розробка системи авторизації та реєстрації користувачів з використанням сучасних методів шифрування даних;

– розробка розділу з новинами для відслідковування та відображення актуальних новин та постів у світі;

– розробка модуля графічного інтерфейсу для відображення маркерів на мапі з інтерактивними можливостями взаємодії для користувачів;

– інтеграція різних модулів в єдиний програмний засіб для забезпечення функціональності порталу та його використання користувачами;

– провести тестування програмного застосунка згідно поставлених задач.

Об’єкт дослідження - процес кінцевого користування та використання онлайн порталу для біженців з персоналізованими рекомендаціями.

Предмет дослідження – є технології, які використовуються для розробки онлайн-порталу.

Методи дослідження. У процесі дослідження використовувались: аналітичний метод - для збору та аналізу даних про потреби біженців та волонтерів, а також для вивчення найкращих практик у сфері розробки веб-порталів; метод проектування - для створення ефективної структури порталу, визначення ключових функцій та властивостей, а також розробки інтерфейсу користувача; метод програмування - для реаліхації програмного застосунку з використанням різних мови програмування. Це включає в себе JSX, React для фронтенду, та Node.js, Express.js для бекенду; метод тестування - для проведення ряду тестів для перевірки функціональності, надійності та зручності використання порталу після його розробки; метод оцінки - для оцінювання ефективності порталу, його відповідності потребам користувачів та можливості для подальшого вдосконалення.

Новизна отриманих результатів: Запропоновано новий підхід до підтримки біженців, інтегруючи інтерактивну мапу, створену за допомогою React-Leaflet, та модуль новин в один онлайн-портал. Це дозволяє надавати користувачам актуальну інформацію та персоналізовані рекомендації на основі їхнього місцезнаходження.

Практичне значення розробленого онлайн порталу полягає у полегшені інтеграції біженців у новому середовищі. Це включає забезпечення доступу до інформації про місцеві ресурси, правову підтримку, освіту, медичне обслуговування та інші необхідні послуги.

Особистий внесок здобувача. Весь внесок у цю роботу був зроблений особисто автором, який включає в себе ідею проекту, його проектування, розробку, тестування, аналіз потреб користувачів та ринку, щоб забезпечити максимальну ефективність та корисність продукту.

Структура та обсяг роботи. Пояснювальна записка до бакалаврської дипломної роботи складається з чотирьох розділів.

У першому розділі розглянуто поточний стан інтегрованих онлайн-порталів для біженців, а також проведено порівняння запропонованого рішення з існуючими системами та сформовано задачі.

У другому розділі надано детальний опис реалізації проєкту, включаючи використані мови програмування, технології та інструменти. Також описано алгоритм та графічний інтерфейс, які використовуються для реалізації програмного забезпечення.

Третій розділ присвячений розробці програмних модулів, необхідних для програмного застосунку. У розділі описані алгоритми та моделі, використані для розробки модулів, а також технології та інструменти, використані в процесі розробки.

У четвертому розділі проведено тестування функціональних та графічних компонентів програмного забезпечення, розробленого в рамках бакалаврської дипломної роботи а також надано керівництво користувача для системи.

Апробація результатів здійснена на конференції XXIV Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів (2024) і опубліковані в тезах доповіді цієї конференції.

Пояснювальна записка містить 59 сторінок тексту, 39 рисунків, 3 таблиці, чотири додатки, 25 найменувань в списку використаних джерел.

1 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ОНЛАЙН-ПОРТАЛУ ДЛЯ БІЖЕНЦІВ З ПЕРСОНАЛІЗОВАНИМИ РЕКОМЕНДАЦІЯМИ

1.1 Аналіз стану наявних онлайн-порталів для біженців.

Сучасний етап розвитку технологій відкриває безліч можливостей для створення та вдосконалення онлайн-порталів для біженців, які стають важливим інструментом для забезпечення доступу до інформації, послуг та підтримки для тих, хто шукає притулок та захист у новому середовищі.

Головні переваги онлайн-порталів для біженців включають доступ до важливої інформації про правовий статус, процедури отримання дозволу на перебування, освітні та медичні послуги, а також можливість спілкування та обміну досвідом між біженцями. Технологічні інновації дозволяють впроваджувати нові функціональності, такі як GPS-навігація та мовні програми для полегшення вивчення місцевих мов. Безпека та конфіденційність на онлайн-платформах є надзвичайно важливою, зокрема захист персональних даних та уникнення кібератак.

Забезпечення доступу до інформації щодо процедур щодо міграції та правового статусу, а також до освітніх та медичних ресурсів, стає ключовою складовою для успішного впорядкування нового життя в іншій країні. Портал, який забезпечує зручний доступ до цієї інформації, може виявитися критичним для біженців у процесі адаптації та інтеграції.

Крім того, можливість спілкування та обміну досвідом між біженцями створює сприятливу атмосферу спільноти, де люди можуть допомагати один одному у складних ситуаціях та ділитися корисною інформацією. Це сприяє психологічній підтримці та створює можливості для соціальної інтеграції.

Зростання доступності до Інтернету та розвиток мобільних технологій дозволяють впроваджувати нові функціональності, які полегшують життя біженців у новому середовищі. Наприклад, GPS-навігація допомагає

зорієнтуватися у новому місті, а мовні програми сприяють вивченню місцевих мов для комунікації та взаємодії з місцевими жителями.

Забезпечення безпеки та конфіденційності на онлайн-платформах є надзвичайно важливою складовою. Біженці, які опиняються у вразливому стані, повинні мати впевненість у захисті своїх персональних даних та конфіденційності своїх комунікацій.

Отже, аналіз стану наявних онлайн-порталів для біженців підкреслює важливість розвитку та вдосконалення цих інструментів для забезпечення підтримки та захисту для біженців.

Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями вимагає комплексного підходу та уваги до потреб цільової аудиторії, а також врахування найновіших тенденцій у сфері технологій та соціальної підтримки.

1.2 Порівняльний аналіз аналогів

Останні роки дійсно принесли значний розквіт у сферах підтримки біженців, особливо в контексті розробки онлайн-порталів. Ці портали відіграють критично важливу роль у наданні біженцям доступу до важливої інформації та ресурсів, які можуть полегшити їх перехід до нового життя.

Онлайн-портали для біженців надають широкий спектр інформації та ресурсів. Вони можуть включати інтерактивні карти, які допомагають біженцям знайти місцеві ресурси, такі як медичні центри, школи, соціальні служби та інші важливі установи. Ці карти можуть бути надзвичайно корисними для біженців, які тільки що прибули в нову країну і ще не знайомі з місцевими умовами.

Крім того, ці портали часто містять сторінки з новинами, які надають актуальну інформацію про політику у сфері біженців, нові законодавчі акти, а також історії успіху інших біженців. Це може бути джерелом надії та натхнення для біженців, які можуть відчувати себе відчайдушно або відчувати себе відчуженими в новому середовищі.

Нижче наведено кілька відомих рішень у цій галузі:

- Online Refugee Aid Portal;
- The Refugee Project;
- Refugee Integration Network;
- Ref Aid;
- Scottish Refugee Council Integration Network.

Одним із прикладів інноваційного використання програмного забезпечення для підтримки біженців є Online Refugee Aid Portal. (рис. 1.1) – це інноваційний програмний засіб, який надає біженцям доступ до важливої інформації [2].

Цей портал допомагає біженцям зрозуміти свій правовий статус, процедури отримання дозволу на перебування, а також надає інформацію про доступні освітні та медичні послуги. Він також сприяє пошуку житла та роботи, що є критично важливим для людей, які пережили переселення. Крім того, цей портал створює можливість для спілкування та обміну досвідом між біженцями, що сприяє створенню сильної спільноти підтримки.

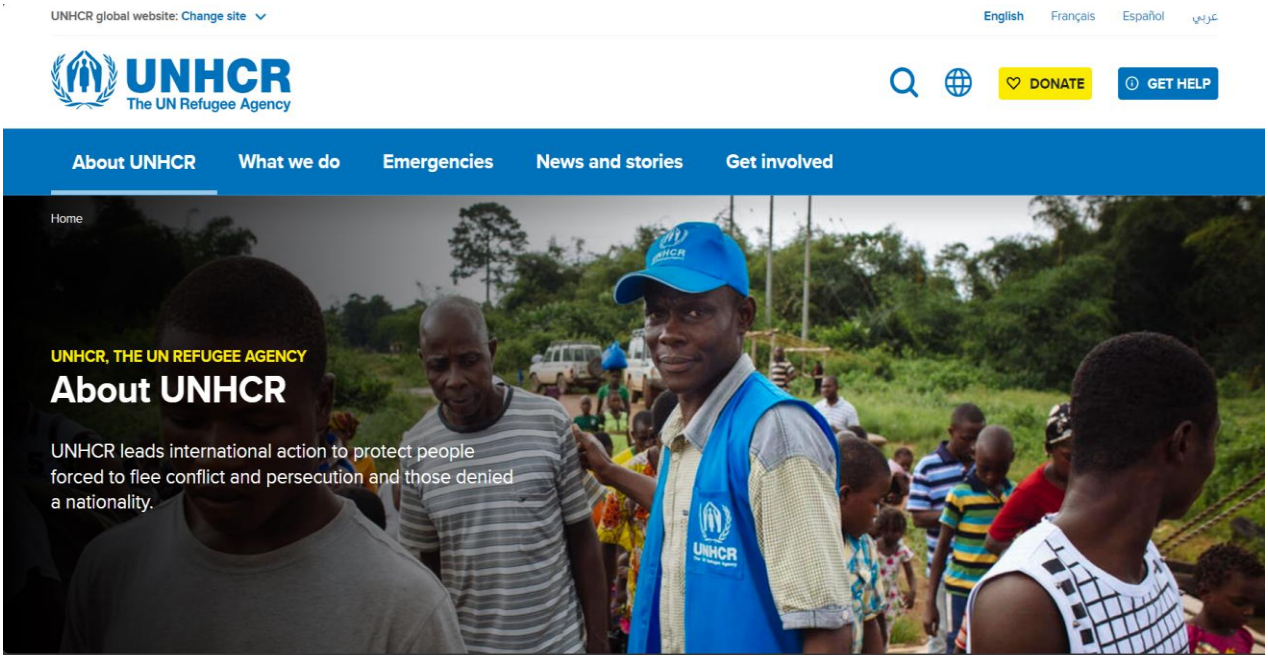

Рисунок 1.1 – Приклад роботи застосунку Online Refugee Aid Portal

Refugee Integration Network (рис. 1.2) - це інноваційний онлайн-портал, створений спеціально для надання біженцям доступу до важливої інформації та ресурсів [3]. Цей портал дозволяє біженцям ознайомитися з різноманітними соціальними програмами, доступними медичними послугами, освітніми можливостями та перспективами зайнятості. Все це робить Refugee Integration Network необхідним інструментом для біженців, які шукають допомогу та підтримку у процесі інтеграції в нове середовище.

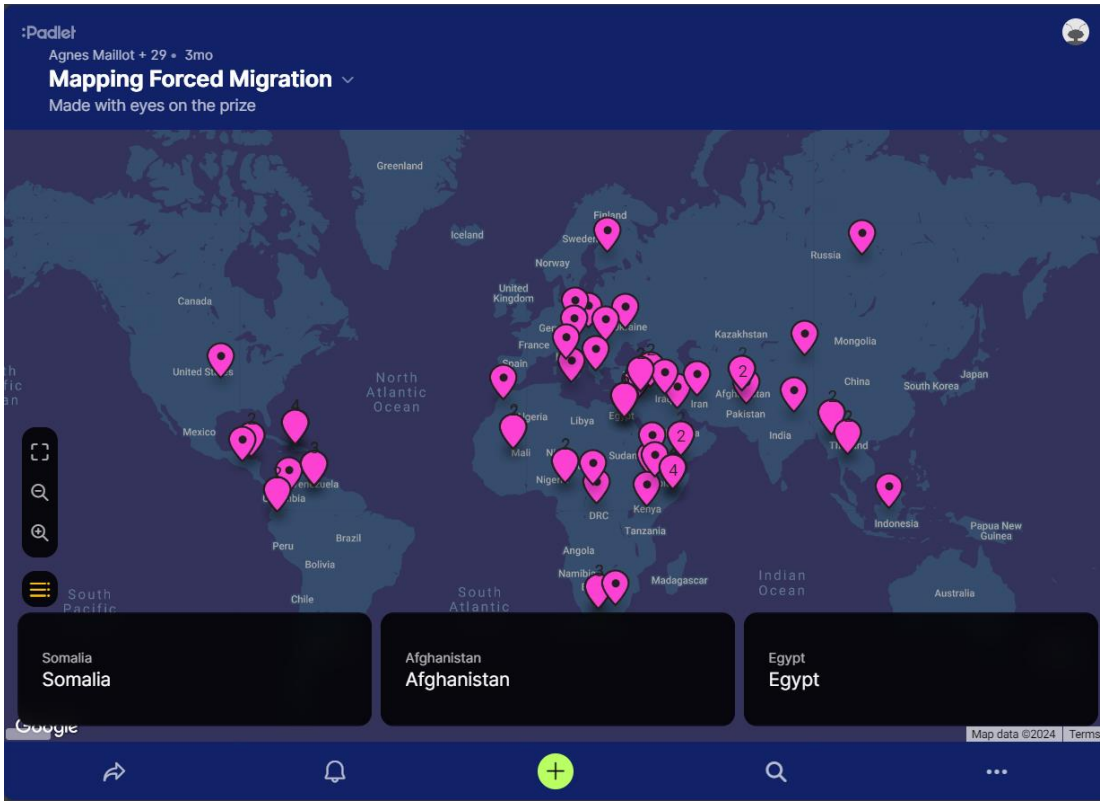

Рисунок 1.2 – Приклад роботи застосунку Refugee Integration Network

Ще одним прикладом є The Refugee Project (рис. 1.3) - потужний застосунок, який візуалізує дані про біженців з 1975 року. Цей проект, розроблений Hyperakt та Ekene Ijeoma, використовує дані УВКБ ООН (Верховного комісара ООН у справах біженців) та дані ООН про населення, щоб розповісти історії про переміщення біженців. Це допомагає зрозуміти, як і чому кризи з біженцями впливали на наш світ протягом останніх півстоліття [4].

Рисунок 1.3 – Приклад роботи застосунку The Refugee Project

RefAid (рис. 1.4) – це мобільний застосунок, який допомагає мігрантам, біженцям та тим, хто їм допомагає, знаходити найближчі послуги за допомогою простого інтерфейсу на карті. Застосунок використовує систему управління контентом та комунікації на основі вебу, яка дозволяє надійним організаціям управляти та оновлювати свої послуги, щоб надавати критично важливу допомогу там, де вона найбільше потрібна [5].

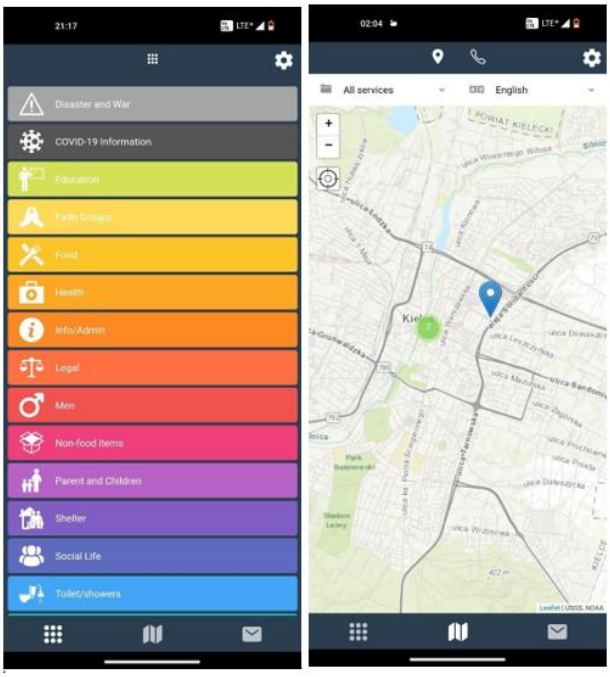

Рисунок 1.4 – Приклад роботи застосунку Ref Aid

Scottish Refugee Council Integration Network (рис. 1.5) - це онлайн-портал, що розташований у південно-східному регіоні Глазго, присвячений підтримці інтеграції громади та відзначенню різноманітності. Складаючись з понад 60 груп учасників, цей портал надає різноманітні послуги на місцевому рівні. Його головною метою є сприяння формуванню міцних зв'язків між громадами приймаючих країн та біженцями, підтримка місцевих ініціатив, надання інформації та сприяння постійному діалогу [6].

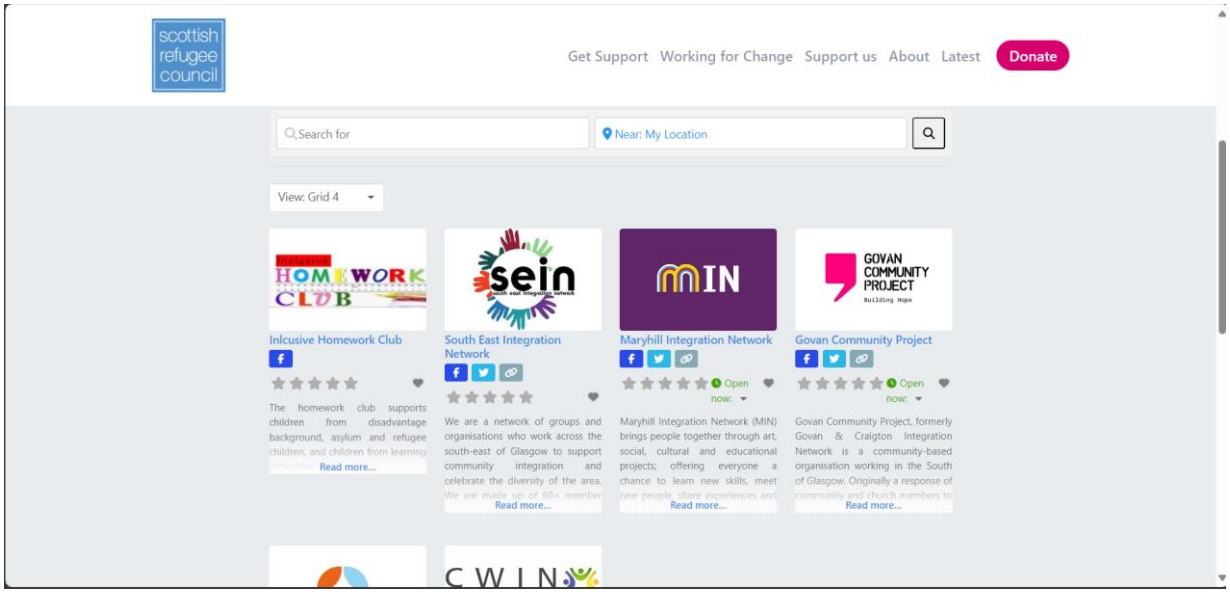

Рисунок 1.5 – Приклад роботи застосунку Scottish Refugee Council Integration Network

Розробка онлайн-порталів для підтримки біженців також вимагає комплексного підходу до програмного забезпечення та дизайну інтерфейсу користувача. Це передбачає створення внутрішньої системи, яка забезпечить доступ до необхідної інформації та ресурсів, полегшить адаптацію та отримання підтримки біженцями з різними рівнями технічної грамотності та можливостями.

Порівняння різних онлайн-порталів для біженців може бути корисним для визначення найкращих рішень для цієї групи користувачів. Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Online Refugee Aid Portal	The Refugee Project	Refugee Integration Network	RefAid	Refugee Council Integration Network	Власний модуль
Доступність інформації та ресурсів	+	+	+	+	+	+
Можливість інтерактивної взаємодії	-	+	+	+	-	+
Підтримка адаптації та отримання підтримки	+	+	+	+	+	+
Швидкість та зручність використання	+	+	+	+	+	+
Модуль з новинами для додаткового отримання інформації	-	-	-	-	-	+
Загальна оцінка	80%	80%	80%	80%	60%	100%

Відповідно до таблиці порівняльних характеристик, розробка власного онлайн-порталу для підтримки біженців може бути доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити розширені можливості для доступу до інформації та ресурсів, що полегшить адаптацію та отримання підтримки біженцями.

Отже такий портал відкриває перед біженцями широкі можливості для отримання необхідної допомоги та інтеграції у нове суспільство.

1.3 Аналіз методів розробки програмного продукту

Оптимальні методи для розв'язання завдань розробки інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями були обрані з урахуванням їхніх переваг та недоліків, розглянемо основні аспекти цього аналізу.

Одним з ключових аспектів успішної платформи є забезпечення зручної та ефективної взаємодії з користувачами. Використання інтерактивної карти та розділ з новинами та постами дозволяє користувачам швидко знаходити ресурси та послуги у відповідних регіонах, сприяючи покращенню користувацького досвіду.

Критично важливим аспектом є захист конфіденційності та особистих даних користувачів. Використання відповідної системи шифрування та відповідність стандартам захисту даних дозволяє зберегти довіру користувачів до платформи.

Реалізація функціоналу для перегляду та створення допомоги для біженців сприяє координації допомоги та підтримує волонтерську діяльність. Це допомагає організувати та забезпечити ефективну допомогу тим, хто найбільше в ній потребує. Узагальнюючи, комбінація різних методів та функціоналу дозволяє створити інтуїтивно зрозумілу та ефективну платформу для підтримки біженців та координації допомоги волонтерів.

Використання бібліотеки use-supercluster у модулі для кластеризації маркерів на мапі дозволяє оптимізувати роботу з великою кількістю даних та поліпшує продуктивність платформи. Ця бібліотека надає можливість об'єднувати близькі маркери на мапі для полегшення навігації користувачів та зменшення навантаження на застосунок.

1.4 Постановка задач інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями

Проаналізувавши переваги та недоліки існуючих онлайн-порталів для біженців, було визначено наступні завдання, які необхідно виконати для успішної розробки програмного забезпечення:

- онлайн-портал має містити два типи облікових записів, для біженців і волонтерів, що забезпечує доступ до різних функціональних можливостей залежно від обраної ролі користувача;
- з метою безпеки даних паролі користувачів порталу мають бути зашифровані;
- зібрана інформація щодо допомоги повинна бути представлена у формі пропозиції, створеної волонтером;
- для відображення місця пропозиції допомоги слід використовувати інтерактивну карту;
- права на управління пропозиціями повинні бути захищені від редагування даних неавторизованими особами, за винятком користувача-творця оферти;
- домашня сторінка порталу повинна мати чіткий і зрозумілий інтерфейс, що забезпечує легкий доступ до функцій користувача;
- провести тестування програмного застосунка згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан наявних онлайн-порталів для біженців, а також був проведений порівняльний аналіз кількох існуючих аналогів для підтримки біженців, таких як Online Refugee Aid Portal, The Refugee Project, Refugee Integration Network, RefAid та Scottish Refugee Council Integration Network. Кожен з цих порталів має свої переваги та особливості, спрямовані на надання доступу до інформації та ресурсів, підтримку адаптації та отримання необхідної допомоги біженцям.

Проаналізувавши переваги та недоліки існуючих програмних засобів, було сформульовано завдання для подальшої розробки власного програмного застосунка для ефективної підтримки біженців.

Ці завдання включають розробку алгоритмів відображення маркерів на інтерактивній карті, роботи кластеризації для відображення маркерів та кластерів на карті та розділ з новинами та постами, де біженці можуть стежити за актуальними подіями у світі для зручного доступу до інформації та ресурсів, необхідних для їхньої соціальної інтеграції та адаптації в новому середовищі.

Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власного програмного застосунку.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз даних для створення інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями

Обробка і аналіз даних є критично важливими етапами в розробці сучасних програмних застосунків, особливо при створенні інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями.

Розпочнемо зі збору різноманітних даних, які містять інформацію про місцезнаходження біженців та пункти надання допомоги. Ці дані можна отримати з різних джерел, таких як бази даних або CSV файли. Після цього географічні дані, які включають точки, лінії та полігональні області, будуть візуалізовані на мапі для зручної інтерактивної взаємодії з користувачем.

Система буде використовувати HTTP протокол та REST API архітектуру для отримання інформації з серверу. REST API є популярним вибором через свою простоту та зрозумілість у виконанні запитів до сервера. Також передбачається можливість переходу на протокол HTTPS для забезпечення безпеки шляхом додаткового шифрування. Це захистить дані від можливих спроб перехоплення або зчитування зловмисниками.

Для оптимізації навігації користувачів і зменшення навантаження на систему, буде застосовано алгоритми, такі як формула гаверсинусів для визначення найближчих пунктів допомоги від поточного місця розташування користувача.

Щодо інформації про користувачів, система отримує обмежений набір даних, який включає ім’я, прізвище, роль (наприклад, статус біженця) та електронну пошту. Ці дані використовуються для ідентифікації користувачів та забезпечення доступу до системи. У разі необхідності додаткової персоналізації в майбутньому, можуть бути розглянуті можливості для збору додаткових персональних даних з користувачів.

Всі ці дані та алгоритми будуть інтегровані в систему за допомогою мови програмування JavaScript, бібліотеки React для фронтенду, Node.js Express.js для бекенду, а також бази даних MongoDB із допомогою бібліотеки Mongoose для зберігання даних.

Розробка програмних модулів інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями є комплексним завданням, яке включає кілька етапів.

По-перше, необхідно створити графічний інтерфейс програмного застосунку, який забезпечить зручну та інтуїтивно зрозумілу взаємодію з системою.

Після цього буде розроблений модуль для імпорту, аналізу та обробки отриманих даних. Для цього планується використання алгоритмів, які дозволять ефективно обробляти інформацію про місцезнаходження користувачів та їхні потреби.

Після обробки даних передбачається створення модулю для візуалізації інформації. Цей модуль дозволить користувачам зручно переглядати та аналізувати надану інформацію про послуги та ресурси для біженців. Для цього планується використання сучасних технологій веб-розробки та візуалізації даних.

Нарешті, буде розроблений модуль для автоматизації процесу групування найближчих пунктів допомоги для біженців на карті. Цей модуль буде використовувати алгоритми кластеризації для групування маркерів на карті, що дозволить біженцям легше орієнтуватися та знаходити необхідні ресурси.

Для реалізації цих модулів будуть використані різноманітні інструменти та бібліотеки програмування, зокрема Webstorm, use-supercluster та react-leaflet.

Отже, розробка програмних модулів інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями на JavaScript, React, Node.js, Express.js та Mongoose є комплексним завданням, яке включає кілька етапів. Кожен модуль повинен бути ретельно спроектований, реалізований та

протестований для забезпечення зручної та ефективної роботи з програмним забезпеченням.

2.2 Розробка інтерфейсу веб-застосунку

При розробці інтерфейсу програмного забезпечення було приділено особливу увагу його зрозумілості та зручності для користувача.

Основними кольорами інтерфейсу обрано чорний та білий, оскільки це колірна палітра, яка асоціюється з формальністю, офіційністю та серйозністю. Допоміжними кольорами обрано помаранчевий та блакитний, оскільки вони контрастують з нашими основними кольорами.

При відкритті програмного додатку, користувача зустрічає головна сторінка з новинами (рис. 2.1).

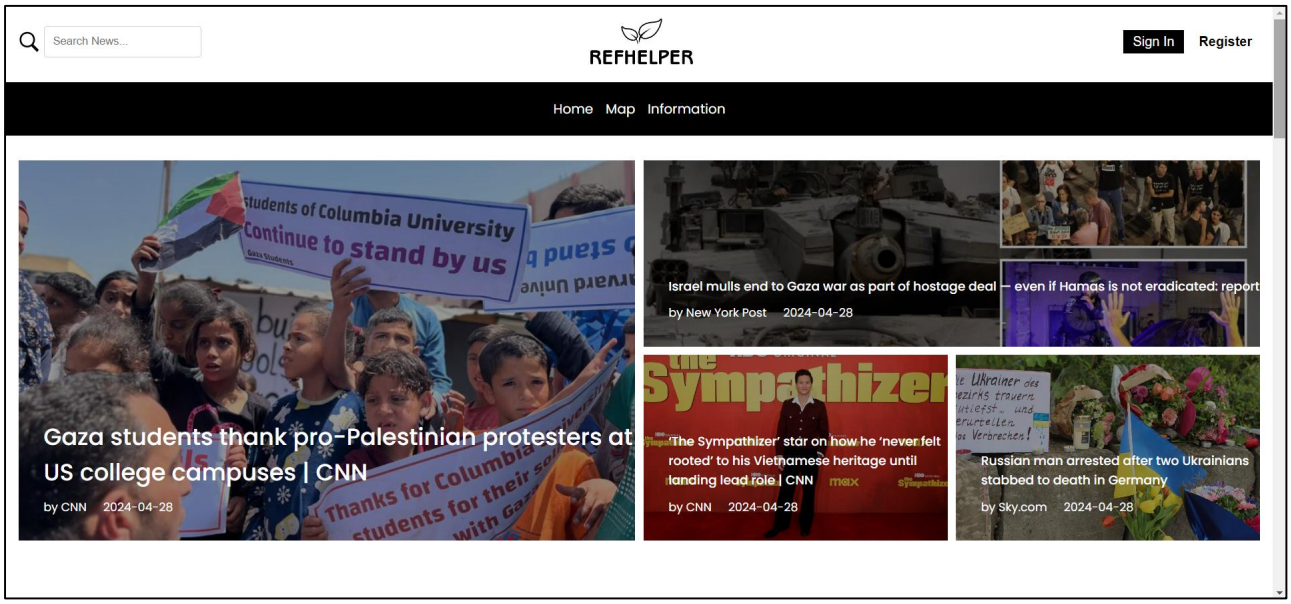

Рисунок 2.1 – Головна сторінка онлайн-порталу

Сторінка з новинами містить актуальну інформацію, яка охоплює широкий спектр тем, від глобальних політичних подій до наукових відкриттів, від культурних заходів до нових технологічних трендів. Це централізоване джерело

новин, що надає користувачам можливість швидко ознайомитися з найновішими подіями та трендами у світі, що дозволяє їм бути в курсі всіх важливих подій.

На цій сторінці кожна тема має своє відведене місце, що сприяє економії часу користувачів, які можуть знаходити важливу інформацію безпосередньо на нашому порталі (рис. 2.2). Це означає, що вони можуть швидко та зручно переглядати новини та статті, які їх цікавлять, не витрачаючи час на перегляд безлічі різних джерел. Такий підхід сприяє зручності користувачів та забезпечує їхню задоволеність від використання нашого порталу.

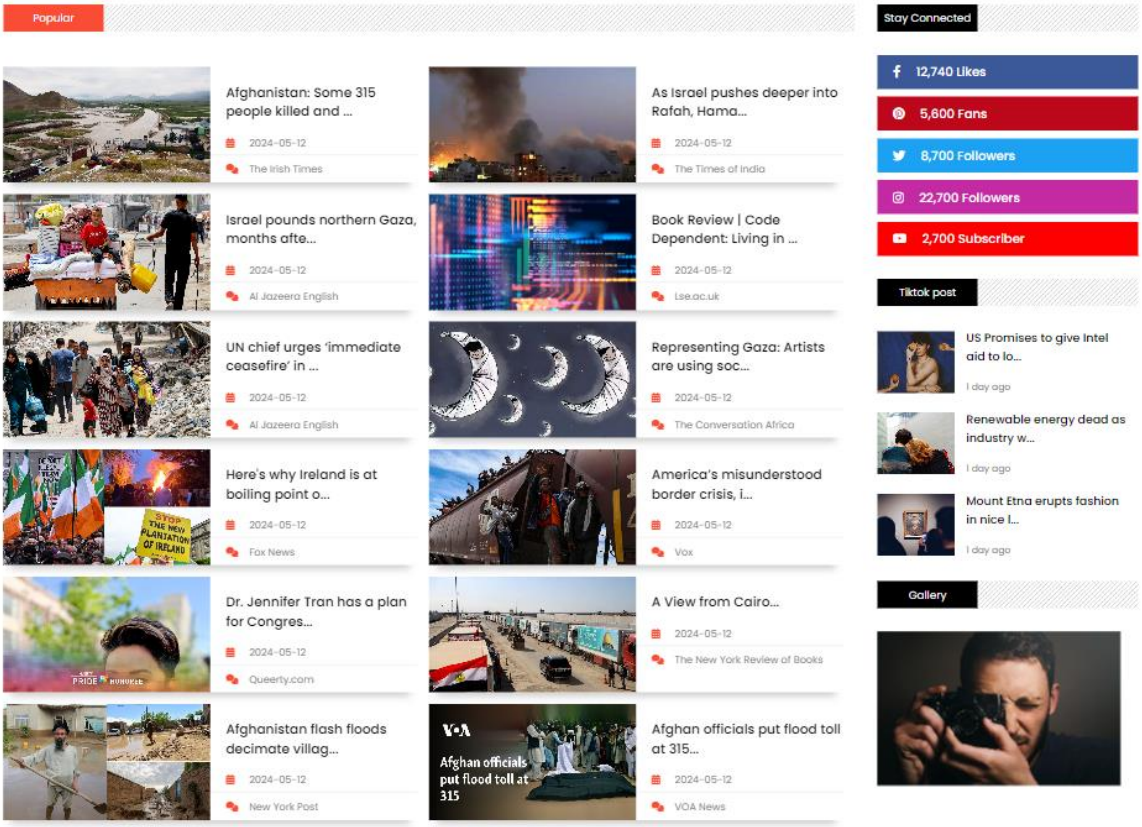

Рисунок 2.2 – Відображення новин в категорії «Popular»

При кліці на заголовок новини чи статті, користувачі можуть переглянути повний текст прямо на нашому порталі, разом із лентою рекомендованих новин та постів зі схожою тематикою (рис. 2.3). Це дозволяє отримувати всю необхідну інформацію без необхідності переходити на інші сайти, що зекономлює час та забезпечує зручність користувачам.

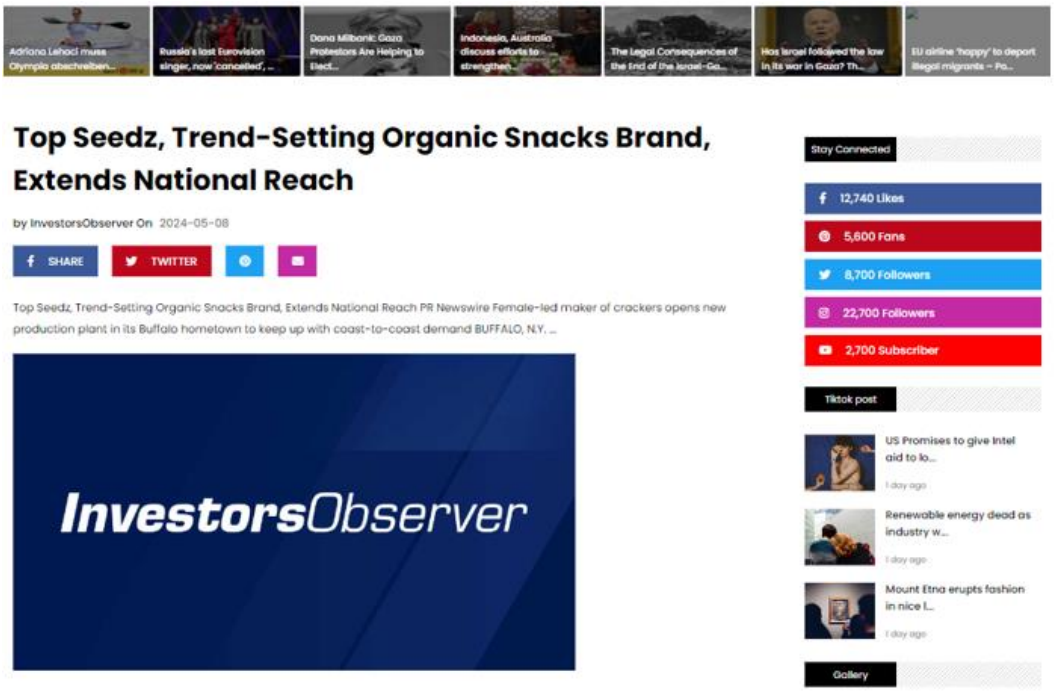

Рисунок 2.3 – Відображення сторінки з новиною

Оскільки онлайн-портал окрім сторінки з новинами передбачає взаємодію з геолокацією користувача й відображенням маркерів та кластерів на карті, я вирішив створити віртуальне робоче середовище з упором на ці можливості. Для цього я розробив інтерактивну карту (рис. 2.4) з маркерами та кластерами, що дозволяє наочно представити цю інформацію користувачеві.

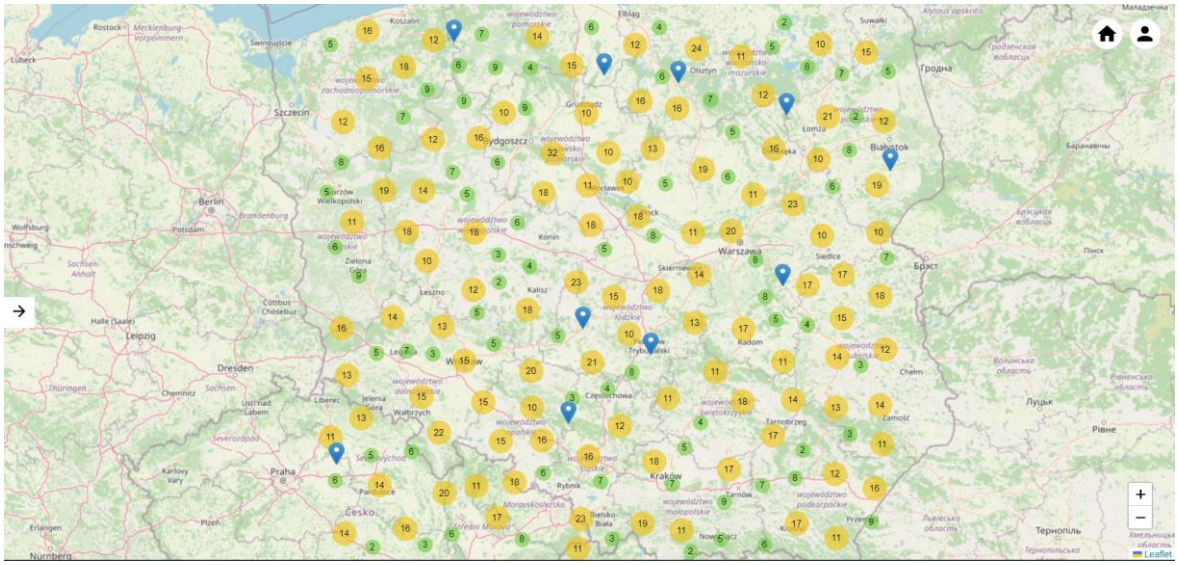

Рисунок 2.4 – Інтерактивна карта застосунку

Незалежно від ролі користувача та його авторизації до аккаунту, сторінка «Map», крім інтерактивної карти, містить бічну панель зі список пунктів допомоги зліва від карти, які відфільтровані за відстанню від місця розташування користувача (рис. 2.5).

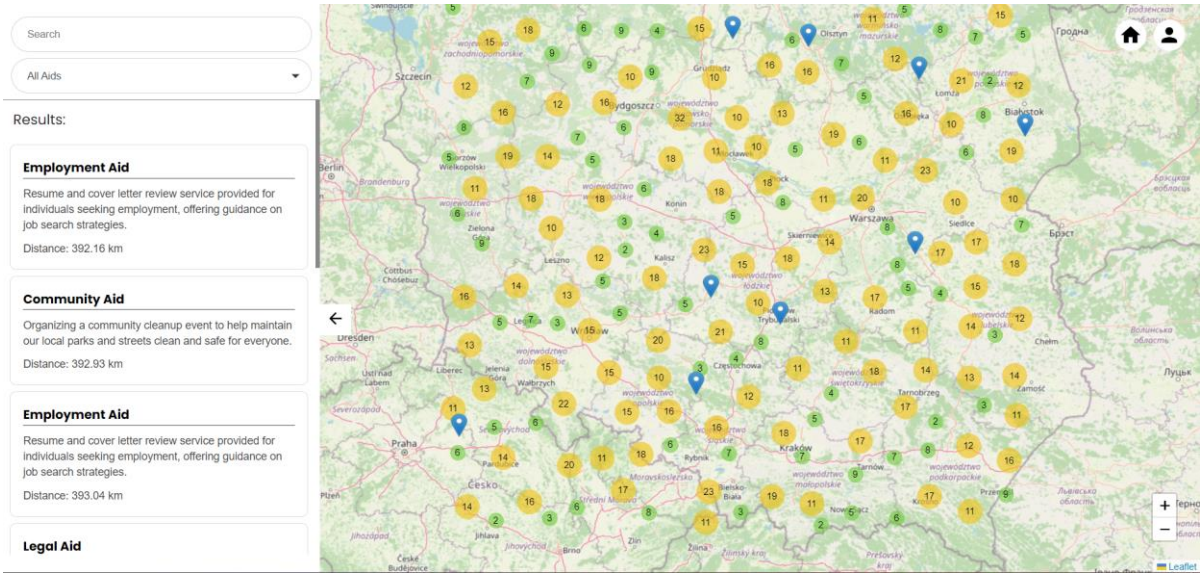

Рисунок 2.5 – Бічна панель інтерактивної карти

Також бічна панель містить пошукову систему (рис. 2.6), яка шукає пропозиції в списку за ключовими словами і фільтром тегів.

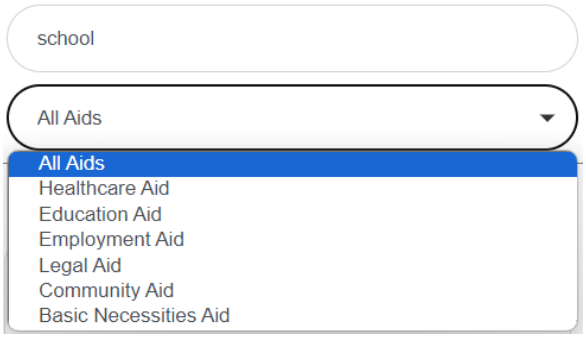

Рисунок 2.6 – Пошукова система панелі

Кожен елемент списку має подію «click», яка переміщує погляд користувача на карті до місця (рис. 2.7), де розташована пропозиція допомоги створена волонтером з повним її описом, адресою та датою створення.

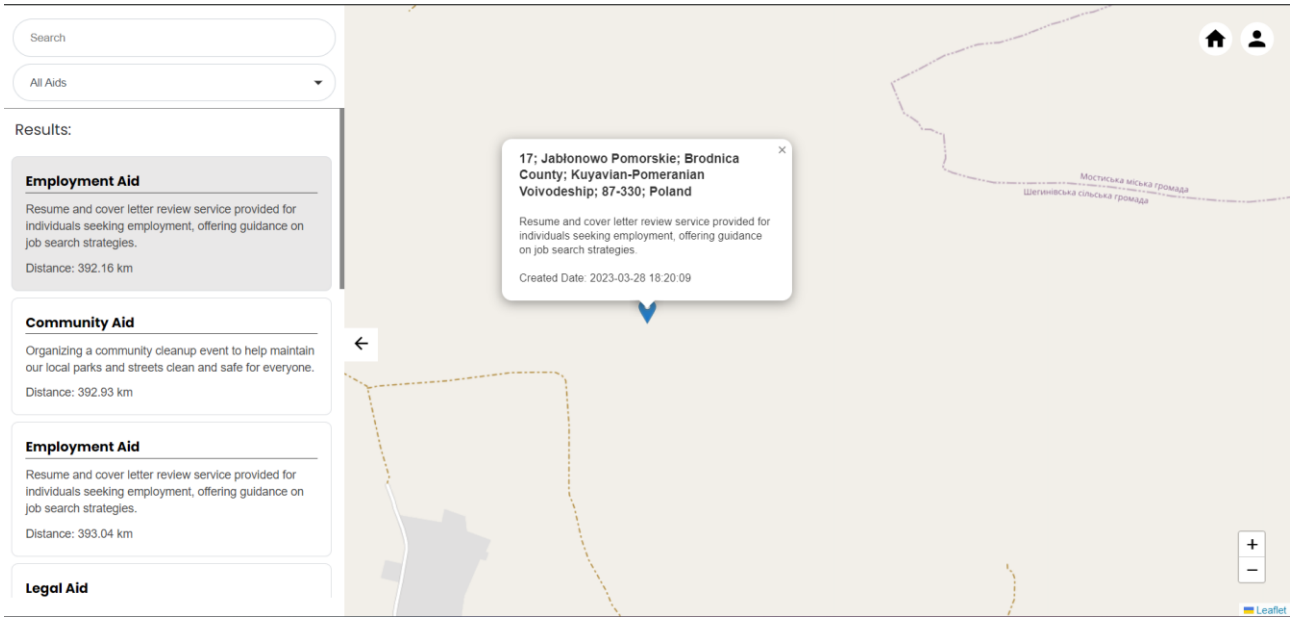

Рисунок 2.7 – Подія click по елементу списка

Портал має можливість реєстрації та авторизації для користувачів, які бажають отримати доступ до додаткових можливостей.

Функція реєстрації дозволяє новим користувачам створювати облікові записи на нашому порталі (рис. 2.8). Під час реєстрації вони мають заповнити обов'язкові поля, такі як ім'я, фамілія, електронна пошта, пароль тощо.

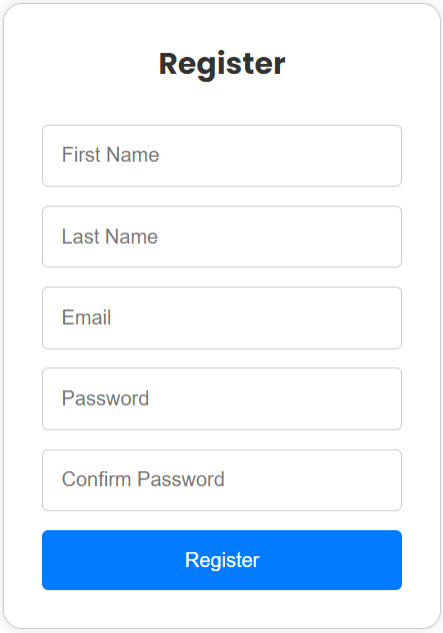

Рисунок 2.8 – Форма реєстрації

Після успішної реєстрації користувач отримує можливість авторизуватися на порталі та користуватися всіма його функціями. Під час реєстрації ми гарантуємо конфіденційність та безпеку особистих даних наших користувачів, використовуючи найсучасніші методи шифрування та забезпечення приватності.

Функція авторизації надає користувачам доступ до особистого облікового запису на нашому порталі (рис. 2.9). Після успішної авторизації вони можуть здійснювати різноманітні дії, такі як розміщення пропозицій допомоги, перегляд своєї історії діяльності та спілкування з іншими користувачами.

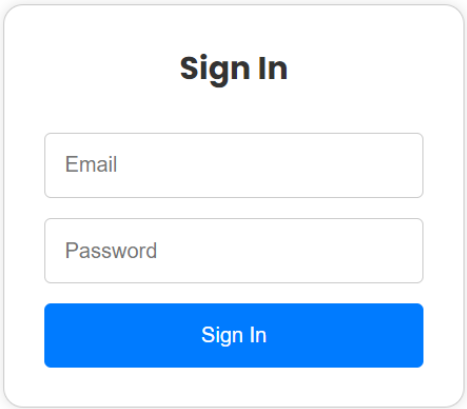

Рисунок 2.9 – Форма авторизації

Процес авторизації забезпечується шляхом введення користувачем свого імені користувача та пароля, після чого система перевіряє правильність введених даних і, у разі успіху, надає доступ до особистого кабінету та відображає електрону адресу користувача у шапці застосунку (рис. 2.10).

Рисунок 2.10 – Успішний вхід до аккаунту

Після успішної авторизації користувач має можливість відкрити свій власний профіль натиснувши на власну електронуадресу (рис. 2.11).

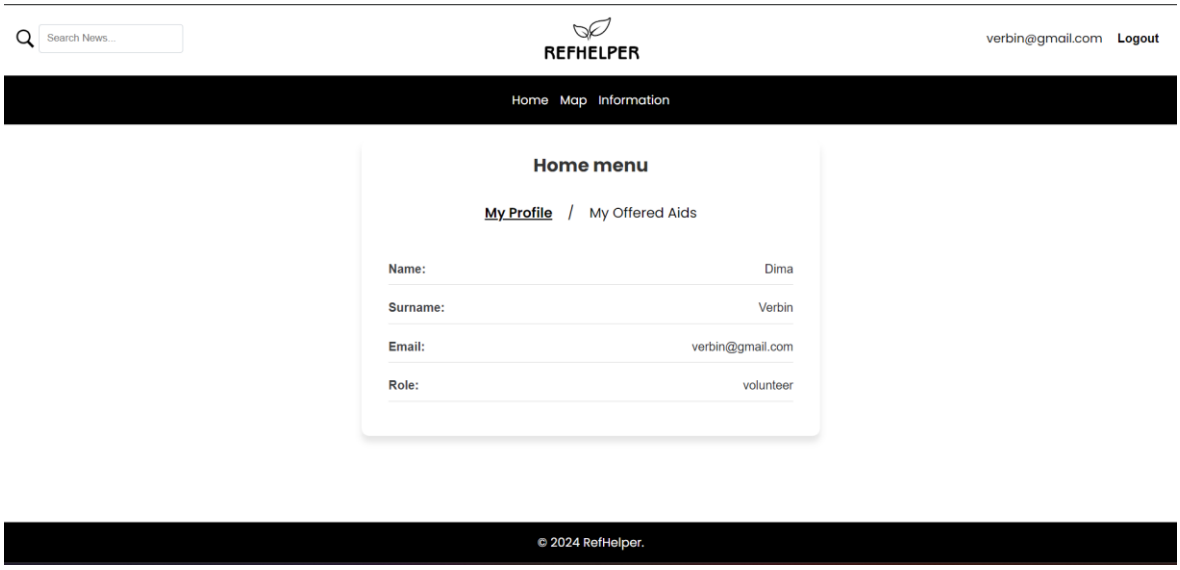

Рисунок 2.11 – Профіль користувача

Інформація та зовнішній вигляд профілю буде залежати від конкретної ролі користувача. Якщо користувач має роль з іменем VOLUNTEER, програма надає йому інформацію про пропозиції, які він створив раніше (рис. 2.12), дає йому можливість створювати нову пропозицію підтримки та можливість оновлювати та видаляти існуючі пропозиції.

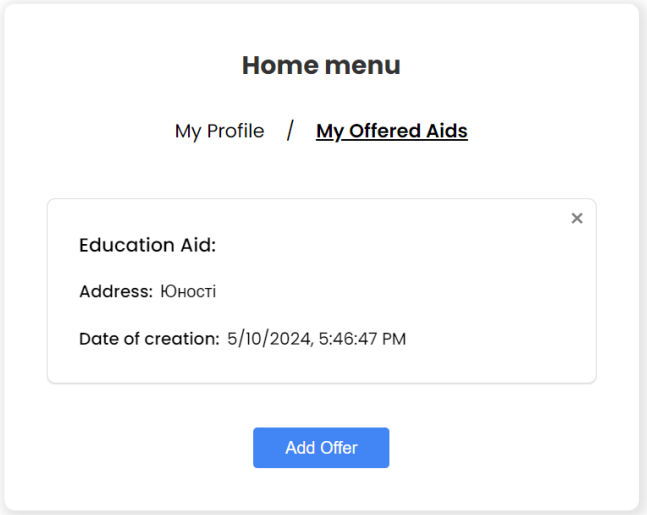

Рисунок 2.12 – Перегляд створених пропозицій з роллю VOLUNTEER

Створення нової пропозиції відбувається при натискані клавіші AddOffer після чого відкривається особиста карта волонтера (рис. 2.13) де відображаються його попередні пропозиції підтримки у вигляді маркерів.

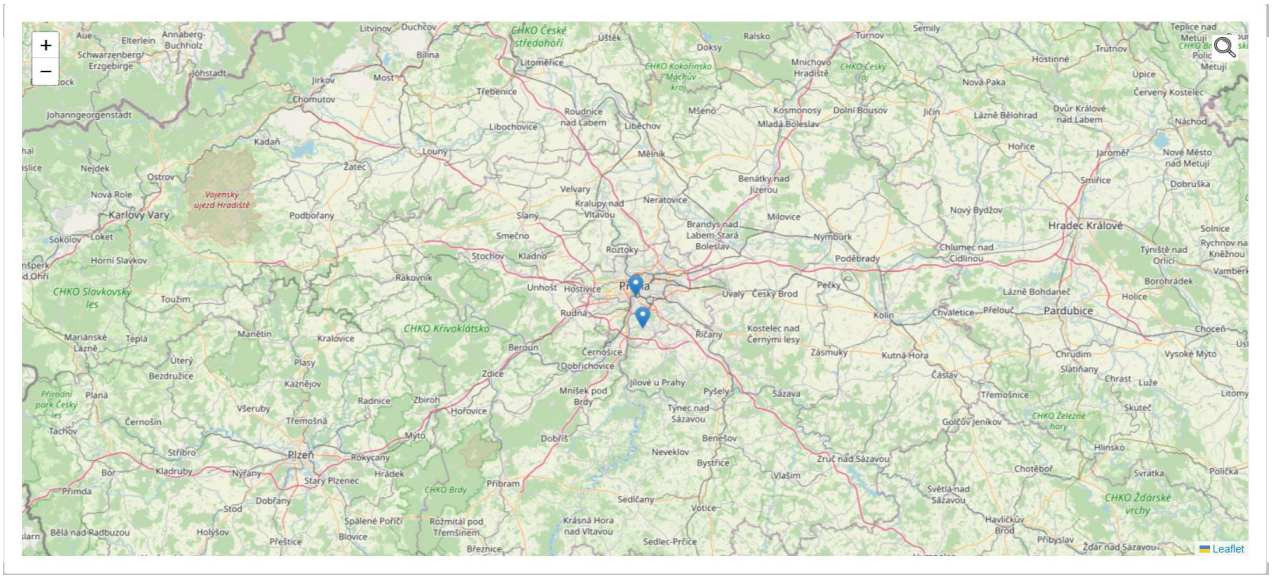

Рисунок 2.13 – Відображення власної карти волонтера з існуючими пропозиціями допомоги

Для створення допомоги, волонтеру потрібно натиснути на потрібне місце на карті після чого відкриється модальне вікно для створення нової пропозиції допомоги (рис 2.14).

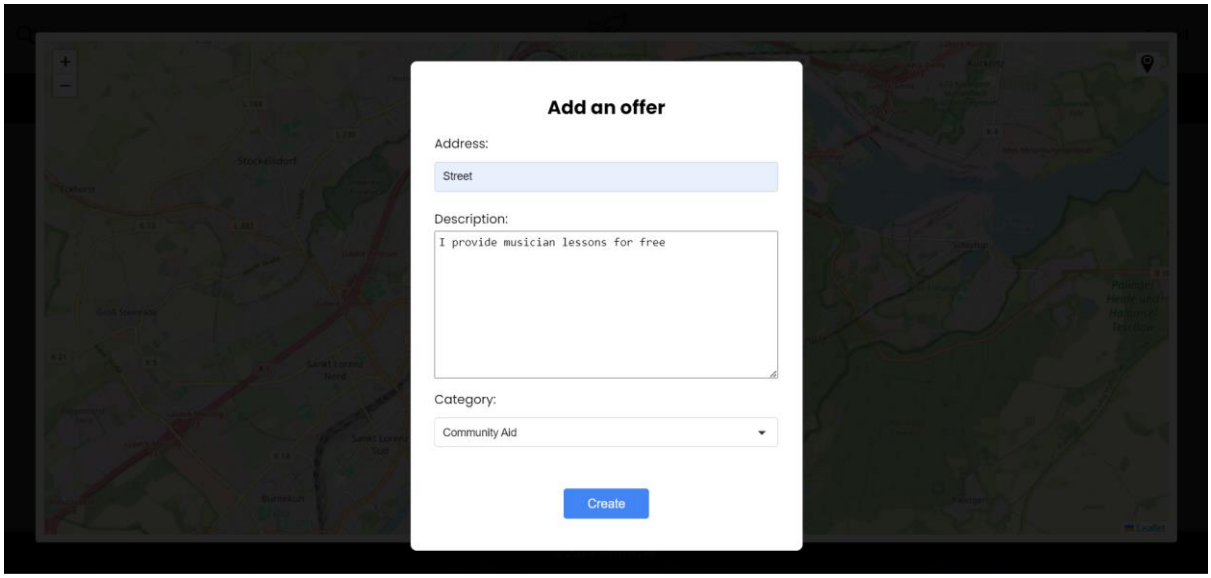

Рисунок 2.14 – Відображення форми створення допомоги

Після введення даних та натискання кнопки «Create» маркер повинен створитися на карті (рис. 2.15).

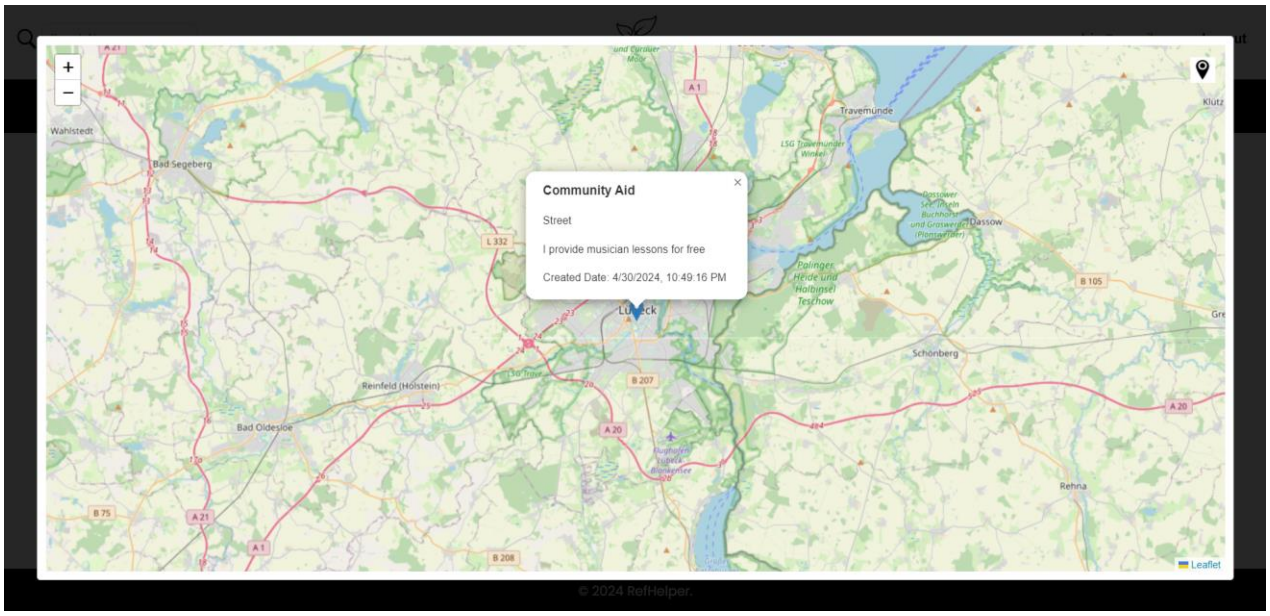

Рисунок 2.15 – Відображення створеної допомоги на карті волонтера

Також допомога з’являється на основній карті порталу, щоб інші користувачі могли її переглянути (рис. 2.16).

Рисунок 2.16 – Відображення створеної допомоги на загальній карті порталу

Якщо волонтер більше не має бажання відображати свою допомогу в нього є можливість зробити видалення створеної пропозиції (рис. 2.17), яка видалиться зі списку історії.

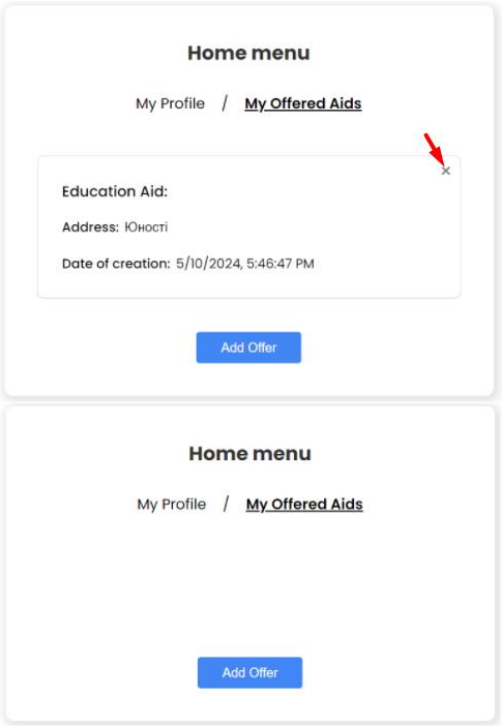

Рисунок 2.17 – Видалення існуючої допомоги з історії волонтера

Видалення повинно відбутися й на загальній карті порталу (рис. 2.18).

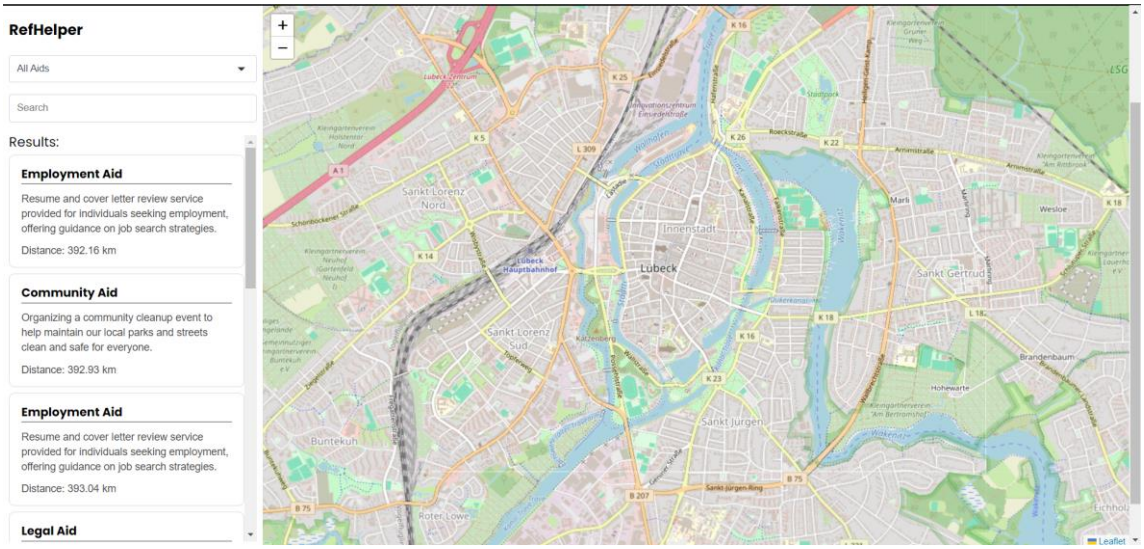

Рисунок 2.18 – Видалення існуючої допомоги із загальної карти

Розробка графічного інтерфейсу програмного застосунку була проведена у середовищі розробки WebStorm, використанням бібліотеки Leaflet, use-superclaster та React.

2.3 Розробка моделі системи

Для кращого розуміння роботи модулів інтегрованого онлайн порталу для біженців з персоналізованими рекмендаціями було вирішено розробити модель роботи системи на прикладі UML діаграми (2.19). Дана діаграма відображає послідовність дій користувачів під час взаємодії з програмним застосунком.

Початково, користувач відкриває програму та переходить до перегляду новин, які надаються на порталі. Після цього він може взаємодіяти з інтерактивною картою, яка відображає розташування точок надання допомоги чи іншу корисну інформацію.

У випадку, якщо користувач бажає зареєструватися на порталі, він має можливість створити новий обліковий запис або увійти в систему, якщо він вже має існуючий.

Після авторизації в системі, користувач може зайти в свій особистий профіль, де він може переглядати свої дані та інформацію, введену під час реєстрації.

Якщо користувач є біженцем, його профіль дозволяє йому переглядати новини та взаємодіяти з інтерактивною картою.

З іншого боку, якщо користувач є волонтером, він отримує додатковий функціонал, включаючи можливість додавання та видалення пропозицій допомоги, а також перегляд своєї історії діяльності на порталі.

Завдяки цій діаграмі діяльності можна краще розібратися в тому, яким чином різні модулі програмного забезпечення порталу взаємодіють між собою та з користувачами, що спрощує процес розробки, тестування та вдосконалення системи.

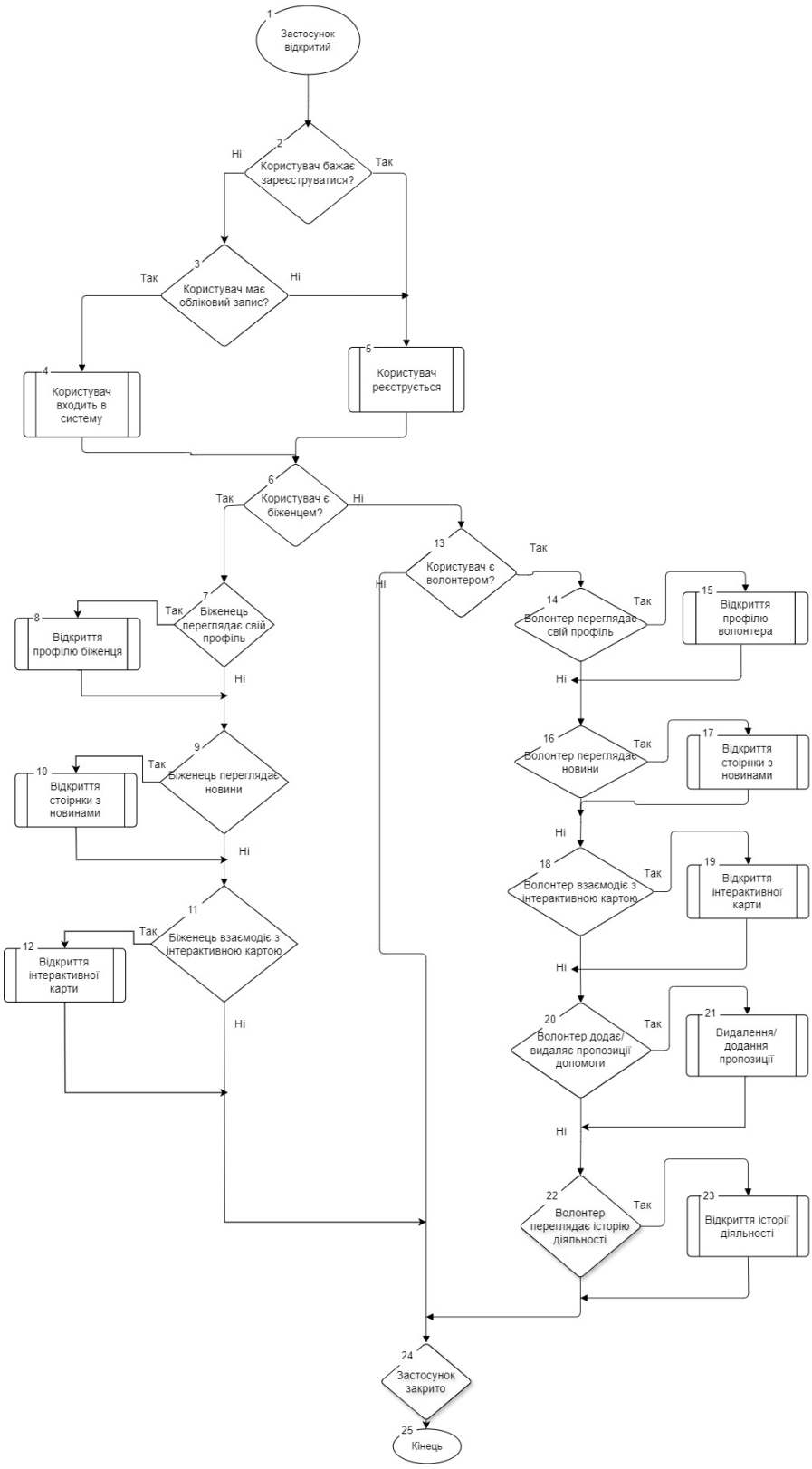

Рисунок 2.19 – UML діаграма програмних засобів

Розробка діаграми діяльності була виконана у програмному забезпеченні LucidChart [7].

2.4 Розробка алгоритмів роботи системи

Для розробки інтерактивної карти інтегровного онлайн-порталу для біженців з персоналізованими рекомендаціями яка є ключовим об’єктом нашого застосунку необхідно розробити загальний алгоритм, згідно якого буде працювати даний програмний модуль (рис. 2.20).

Ця діаграма відображає алгоритм роботи програмного засобу для відображення маркерів допомоги на інтерактивній карті. Згідно даного алгоритму, процес створення інтерактивної карти в компоненті Map починається з імпорту необхідних бібліотек та компонентів.

Після цього створюється функція fetchMarkers, яка відповідає за отримання маркерів з сервера. Ця функція використовує fetch для надсилання запиту до сервера і повертає відповідь у форматі JSON.

Наступним кроком є створення компонента Map. В цьому компоненті ініціалізується стан, який включає місцезнаходження користувача, обрану категорію допомоги, текст пошуку, поточну сторінку, обрані координати маркера та категорії допомоги.

Далі визначається функція getLocation, яка відповідає за отримання місцезнаходження користувача.

Якщо геолокація підтримується браузером, ця функція отримує місцезнаходження користувача і зберігає його в стані.

Потім використовується useQuery для отримання маркерів з сервера. Ця функція використовує функцію fetchMarkers, яку ми визначили раніше.

Маркери, отримані від сервера, фільтруються за обраною категорією допомоги та текстом пошуку. Потім для кожного маркера обчислюється відстань до користувача.

Після цього відображаються перші 10 маркерів. Якщо місцезнаходження користувача доступне, відображається карта з маркерами. Якщо місцезнаходження користувача недоступне, відображається повідомлення “Loading…”.

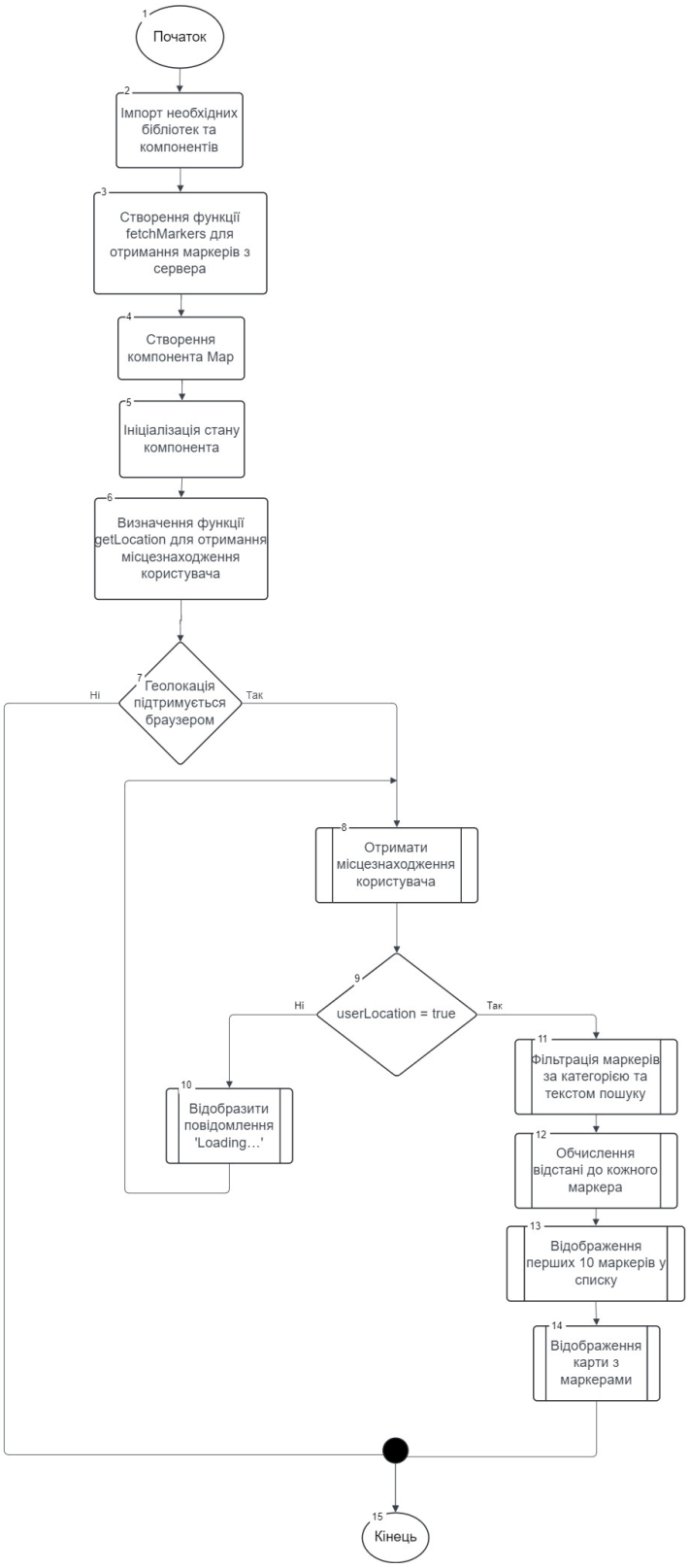

Рисунок 2.20 – Алгоритм роботи відображення маркерів на інтерактивній карті

Для кращої роботи роботи програмного засобу для відображення маркерів допомоги на інтерактивній карті було розроблено додатковий лгоритм

кластеризації для групування маркерів задля полегшення навігації користувачів та зменшення навантаження на застосунок (рис. 2.21). процес починається з імпорту необхідних бібліотек та компонентів. Потім створюється компонент ShowCrimes, в якому визначаються іконки для маркерів. Ці іконки використовуються для відображення маркерів на карті в залежності від того, чи є вони частиною кластера. Використовується алгоритм кластеризації для групування маркерів, які знаходяться близько один до одного на карті. Це допомагає зробити відображення маркерів на карті більш читабельним, особливо коли маркерів багато.

Ми використовуємо бібліотеку use-supercluster для кластеризації. Ця бібліотека приймає масив точок (ваші маркери), область видимості карти (bounds), поточний масштаб карти (zoom) та деякі інші параметри, і повертає масив кластерів для відображення на карті.

Кожен кластер може бути або окремим маркером, або групою маркерів. Це визначається властивістю cluster кожного об’єкта кластера. Якщо cluster дорівнює true, це означає, що кластер складається з декількох маркерів, які згруповані разом. Якщо cluster дорівнює false, це означає, що кластер насправді є окремим маркером.

Для кожного маркера (clusters) виконується перевірка: якщо маркер є кластером, відображається значок кластера; якщо маркер не є кластером, відображається значок маркера.

Далі відбувається обробка подій клацання по маркеру. Коли користувач клацає по кластеру, карта збільшує масштаб до цього кластера, дозволяючи користувачу бачити окремі маркери, які утворюють цей кластер. Якщо користувач клацає по окремому маркеру, відображається спливаюче вікно з описом цього маркера.

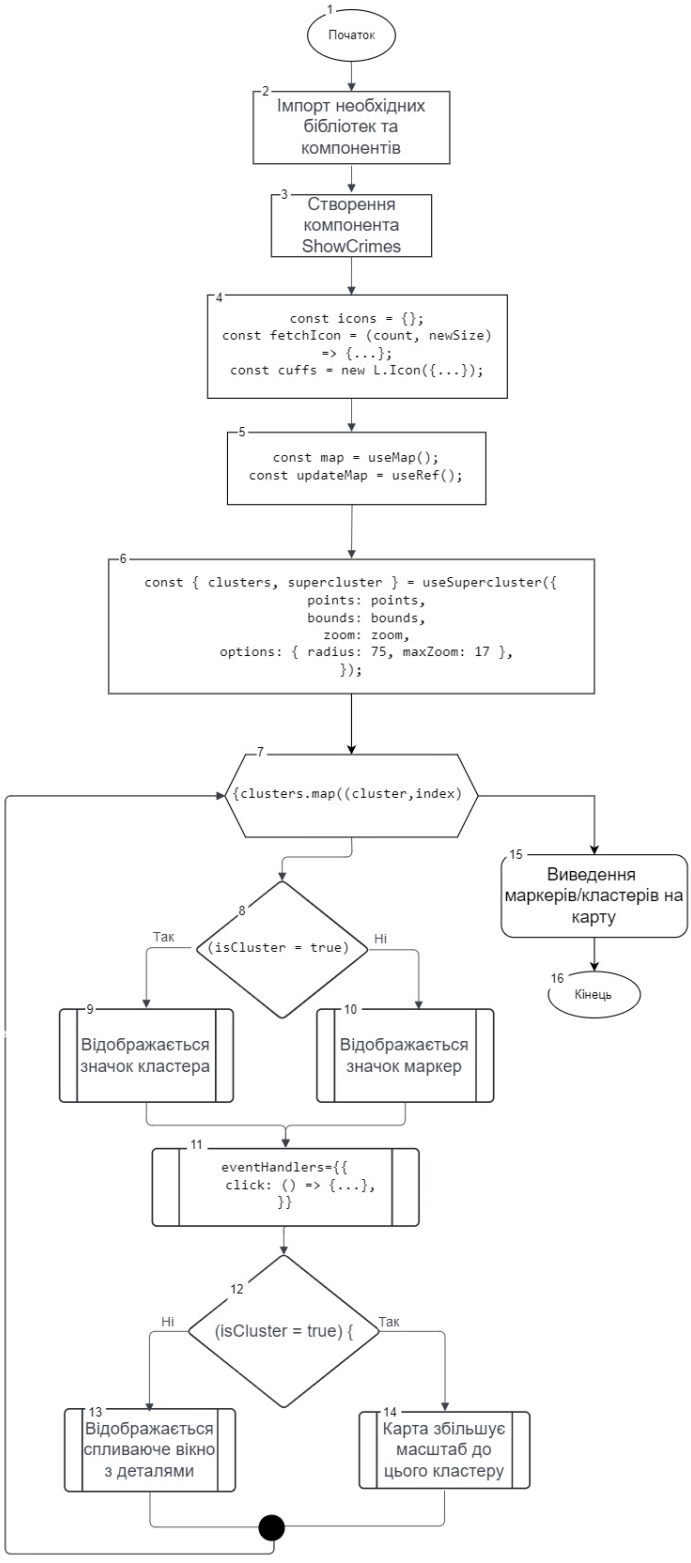

Рисунок 2.21 – Алгоритм роботи кластеризації маркерів

Цей алгоритм кластеризації допомагає забезпечити чистий та організований вигляд карти, незалежно від кількості маркерів, які потрібно відобразити.

2.5 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних програмних засобів. Також було розглянуто процес створення графічного інтерфейсу програмного застосунку. Було розроблено основний алгоритм роботи програмних засобів та алгоритм кластеризації для відображення маркерів та кластерів на карті. Також було розроблену модель роботи програмного продукту за допомогою UML діаграм.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ІНТЕГРОВАНОГО ОНЛАЙН-ПОРТАЛУ ДЛЯ БІЖЕНЦІВ З ПЕРСОНАЛІЗОВАНИМИ РЕКОМЕНДАЦІЯМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації веб-застосунку

При розробці програмних засобів реалізації онлайн порталу для біженців важливо обрати найбільш оптимальні технології для розробки кожного модуля програмних засобів. Згідно даного алгоритму, для розробки порталу було використано набір сучасних технологій та інструментів, що дозволили створити ефективний та надійний продукт.

Основною мовою програмування було обрано JavaScript, відому як динамічну, об'єктно-орієнтовану мову, що використовує прототипи. Вона реалізує стандарт ECMAScript і зазвичай застосовується для створення сценаріїв вебсторінок. Це дозволяє клієнтській частині взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером і змінювати структуру та зовнішній вигляд вебсторінки [8]. Тому серед її основних фреймворків було обрано Angular, Vue.js та React.

Angular – [9] це популярний фреймворк з відкритим вихідним кодом, створений командою Google. Він використовує TypeScript, HTML та CSS для розробки веб-додатків. Серед ключових характеристик Angular є компонентна архітектура, прив'язка даних, маршрутизація, сервіси та можливість тестування. Angular має активну спільноту розробників і пропонує доступ до багатьох корисних бібліотек та інструментів.

Vue.js – [10] це сучасна JavaScript-бібліотека, яка допомагає створювати веб-інтерфейси та односторінкові додатки (SPA). Вона відрізняється декларативним стилем програмування і компонентною архітектурою. Vue.js відомий своєю простотою у вивченні та використанні, а також високою продуктивністю, що робить його надзвичайно популярним серед розробників.

Основні характеристики Vue.js включають легкість освоєння, використання компонентів, прив'язку даних, директиви, модульність та реактивність.

React – [11] це JavaScript-бібліотека, створена Facebook для розробки користувацьких інтерфейсів. Вона використовує віртуальний DOM для оптимізації продуктивності додатків і може застосовуватися як окремо, так і разом з іншими технологіями, такими як Redux для управління станом. Ключові характеристики React включають архітектуру на основі компонентів, віртуальний DOM, односторонній потік даних, синтаксис JSX, можливість розширення та велику спільноту розробників.

Після аналізу цих фреймворків та бібліотек було складено таблицю, яка демонструє відмінності між Angular, Vue.js та React. Результати відмінностей зведено в табл. 3.1.

Таблиця 3.1 – Функціональні характеристики бібліотек та фреймворків

Характеристика	React	Angular	Vue.js
Швидкість рендерингу	+	-	+
Компонентний підхід	+	+	-
Розмір бібліотеки	+	-	+
Документація та підтримка	+	-	+
Розширюваність та відкритість для сторонніх бібліотек	+	-	+
Легкість навчання та використання	+	-	+
Сума	6	1	5

Проаналізувавши дані в таблиці, можна зробити висновок, що React є найкращим вибором для розробки фронтенд-модулів, перевершуючи Angular та Vue за швидкістю рендерінгу та методологією розробки.

Є декілька середовищ розробки для створення веб-додатків за допомогою JavaScript, в нашому випадку TypeScript для типізації. Для порівняння були обрані “Visual Studio Code”, “Webstorm” і “Atom”.

Visual Studio Code (VS Code) – [12] це безкоштовний редактор коду від Microsoft, який здобув популярність серед розробників завдяки своїм потужним можливостям і розширенням, що спрощують процес програмування. Головні особливості VS Code включають підтримку різних платформ, зручність використання, підсвічування синтаксису, функцію автодоповнення, інтеграцію з Git, розширення та інструменти для відладки.

WebStorm – [13] це інтегроване середовище розробки (IDE) для веб-програмування від компанії JetBrains. Воно зосереджене на створенні веб-додатків за допомогою технологій, таких як HTML, CSS, JavaScript, а також фреймворків і бібліотек, таких як React, Angular, і Vue.js. Ключові характеристики WebStorm включають редактор коду, підтримку різних мов програмування та технологій, можливості для відладки, управління проектами, автоматизацію завдань і широкий спектр розширень.

Atom – [14] це безкоштовний текстовий редактор з відкритим кодом, розроблений GitHub. Його можна використовувати для редагування та написання різноманітних типів файлів, включаючи програмний код та веб-сторінки. Основні особливості Atom включають можливість розширення функціоналу, зручний інтерфейс користувача, підсвічування синтаксису, функції пошуку та заміни, керування проектами та інтеграцію з Git.

Для візуального порівняння цих середовищ розробки було створено таблицю порівняння (таблиця. 3.2).

Таблиця 3.2 – Функціональні характеристики середовищ розробки

Характеристика	VS Code	WebStorm	Atom
Відладка коду	-	+	-

Продовження таблиці 3.2

Підтримка мов програмування	-	+	+
Швидкість	+	-	-
Наявність плагінів та розширень	+	+	-
Надійність та стабільність	+	+	-
Сума	3	4	1

Проаналізувавши дані з таблиці можна підсумувати, що WebStorm найкраще підходить для розробки веб-проєкту на мові JavaScript, адже саме ця IDE має у собі максимальні можливості для програмування та тестування коду.

Для відображення інтерактивних карт використовувалася бібліотека Leaflet.

Leaflet - [15] це потужна бібліотека JavaScript для створення інтерактивних карт, яка відзначається легкістю використання. За допомогою Leaflet можна легко відображати маркери, шари та інші елементи на карті, а також обробляти події.

Для взаємодії з базою даних у проєкті використовувалася бібліотека Mongoose. Mongoose - [16] це інструмент для Node.js, який забезпечує зручний інтерфейс для роботи з MongoDB. Він спрощує процес створення, зчитування, оновлення та видалення даних.

MongoDB - [17] це відкрита система управління базами даних, яка базується на документах. Вона не вимагає строго визначеної схеми таблиць. MongoDB займає проміжне положення між системами, що оперують даними у форматі ключ-значення, та реляційними базами даних, пропонуючи швидкодію і масштабованість разом із зручністю формування запитів.

Серверна частина порталу була розроблена з використанням Node.js та Express.js.

Node.js - [18] це середовище виконання JavaScript, яке дозволяє виконувати JavaScript-код на серверній стороні.

Express.js - [19] - це мінімалістичний та гнучкий веб-серверний фреймворк для Node.js, який надає широкий набір потужних функцій для розробки веб-додатків.

Усі етапи розробки були виконані в середовищі WebStorm, яке надає потужні інструменти для розробки веб-додатків, включаючи автозавершення коду, налагоджувач, вбудований термінал та інші.

Для розрахунку відстані між двома точками на поверхні Землі використовувалася формула гаверсинусів [20]. Ця формула дозволяє точно визначити відстань між точками, враховуючи сферичну форму Землі.

Для групування маркерів, які знаходяться близько один до одного на карті, застосовувався алгоритм кластеризації [21]. Цей алгоритм дозволяє об'єднувати маркери в кластери, що полегшує візуалізацію та аналіз даних.

Ці технології та інструменти були обрані з огляду на їхню надійність, продуктивність та широке застосування в індустрії веб-розробки. Вони забезпечують гнучкість, швидкість та ефективність при розробці складних веб-додатків, таких як цей портал.

Поєднання цих інструментів надає комплексний підхід до розробки, що дозволяє створювати продукт з високим рівнем функціональності, надійності та зручного користування.

3.2 Розробка модуля кластеризації маркерів

При розробці модуля кластеризації маркерів важливо враховувати ряд технічних аспектів. При розробці модуля для кластеризації маркерів на мапі, першим кроком є імпорт необхідних бібліотек та компонентів. Для цього використовуються React, use-supercluster та react-leaflet.

Після імпорту бібліотек, наступним кроком є створення іконок для маркерів на мапі. Для цього використовується функція fetchIcon, яка генерує іконки в залежності від кількості маркерів у кластері.

Наступним кроком є створення основного компонента ShowCrimes, який відповідає за відображення маркерів на мапі. Він приймає дані про кластер та координати вибраного маркера як вхідні параметри (рис. 3.3). Всередині цього компонента, використовується useEffect для оновлення мапи, коли вона змінює своє положення або зум. Коли мапа зупиняє рух, викликається updateMap.current() для оновлення границь та зуму мапи.

Після цього, створюється масив points, де кожен маркер перетворюється на об’єкт з геометрією та властивостями, які використовуються для створення кластерів.

За допомогою useSupercluster ми створюємо кластери з маркерів. Він приймає масив points, границі мапи, зум та опції для створення кластерів.

Наступний фрагмент коду використовує useEffect для встановлення виду мапи на вибрані координати маркера. Коли selectedMarkerCoords змінюється, мапа автоматично переміщується до нових координат з анімацією.

На останньому етапі, ми відображаємо маркери та кластери на мапі. Для кожного кластера створюється маркер на мапі. Якщо кластер містить більше одного маркера, він відображається як кластер. В іншому випадку він відображається як окремий маркер.

3.3 Розробка модуля для відображення маркерів на інтерактивній карті

Головною задачею модуля графічного інтерфейсу для відображення маркерів є створення інтерактивної мапи, яка відображає маркери з описом допомоги. Для цього було використано бібліотеки React, react-query та react-leaflet.

Після імпорту бібліотек та компонентів, ми створюємо функцію fetchMarkers, яка використовується для отримання даних про маркери з сервера. Вона виконує запит до сервера та повертає відповідь у форматі JSON.

Наступним кроком є створення компонента Map, який відповідає за відображення мапи та маркерів на ній. Він використовує стан для зберігання різних значень, таких як поточне місцезнаходження користувача, вибрану допомогу, текст пошуку, поточну сторінку, вибрані координати маркера та категорії.

Всередині цього компонента, ми використовуємо useEffect для отримання поточного місцезнаходження користувача. Якщо браузер підтримує геолокацію, він отримує поточні координати користувача та зберігає їх у стані.

Далі ми використовуємо useQuery для отримання даних про маркери з сервера. Ми використовуємо функцію fetchMarkers, яку ми створили раніше, для виконання запиту до сервера.

Після отримання даних, ми фільтруємо маркери на основі вибраної допомоги та тексту пошуку. Якщо вибрана допомога є “all”, ми повертаємо всі маркери. В іншому випадку ми повертаємо тільки маркери, які відповідають вибраній допомозі та тексту пошуку.

Наступним кроком є розрахунок відстані від поточного місцезнаходження користувача до кожного маркера. Ми використовуємо функцію distanceTo з бібліотеки leaflet для розрахунку відстані в метрах, а потім перетворюємо її в кілометри. Також відбувається сортування елементів по їх відстані від місця розташування користувача.

На останньому етапі, ми відображаємо маркери на мапі. Ми включаємо елементи інтерфейсу для пошуку та фільтрації маркерів, а також відображаємо мапу з маркерами. Якщо користувач вибирає маркер зі списку, мапа автоматично переміщується до цього маркера.

3.4 Інтеграція модулів та публікація на npm

У цьому пункті описано процес інтеграції розроблених модулів, налаштування необхідних пакетів, конфігурації package.json та babel, а також процес транспіляції коду в папку dist і публікації результату на npm.

Ініціалізація проекту: Спочатку було виконано команду npm init для створення файлу package.json проекту (рис. 3.1). Цей файл містить інформацію про проект та його залежності.

Розробка та інтеграція модулів: Після розробки модулів кластеризації маркерів та відображення маркерів на інтерактивній карті, ці модулі були інтегровані в один модуль. Це включало в себе імпорт модулів у компоненти React та їх використання для створення інтерактивної карти.

Налаштування пакетів: Було встановлено та налаштовано всі необхідні пакети для проекту, які вказані в package.json. Це включає в себе бібліотеки React, use-supercluster, react-leaflet, @emotion/react, @emotion/styled, @mui/icons-material, @mui/material, @mui/styled-engine-sc, react-dom, react-scripts, styled-components, web-vitals для залежностей проекту та @babel/cli, @babel/core, @babel/preset-env, @babel/preset-react, babel-loader, css-loader, leaflet, react-router-dom, style-loader для залежностей розробки.

Налаштування Babel: Babel був налаштований для транспіляції коду, додавши @babel/preset-react та @babel/preset-env до файлу .babelrc. Це дозволяє Babel перетворювати JSX та ES6 код в код старого формату, який може виконуватися в більшості браузерів.

Транспіляція коду: Було виконано команду npm run build, яка використовує Babel для транспіляції коду з директорії src до директорії dist. Це перетворює JSX та ES6 код в код , який може виконуватися в більшості браузерів.

Публікація на npm: Нарешті, проект було опубліковано на npm. Для цього було виконано команду npm login для входу в обліковий запис npm. Після цього, було введено команду npm publish для публікації проекту на npm (рис. 3.1).

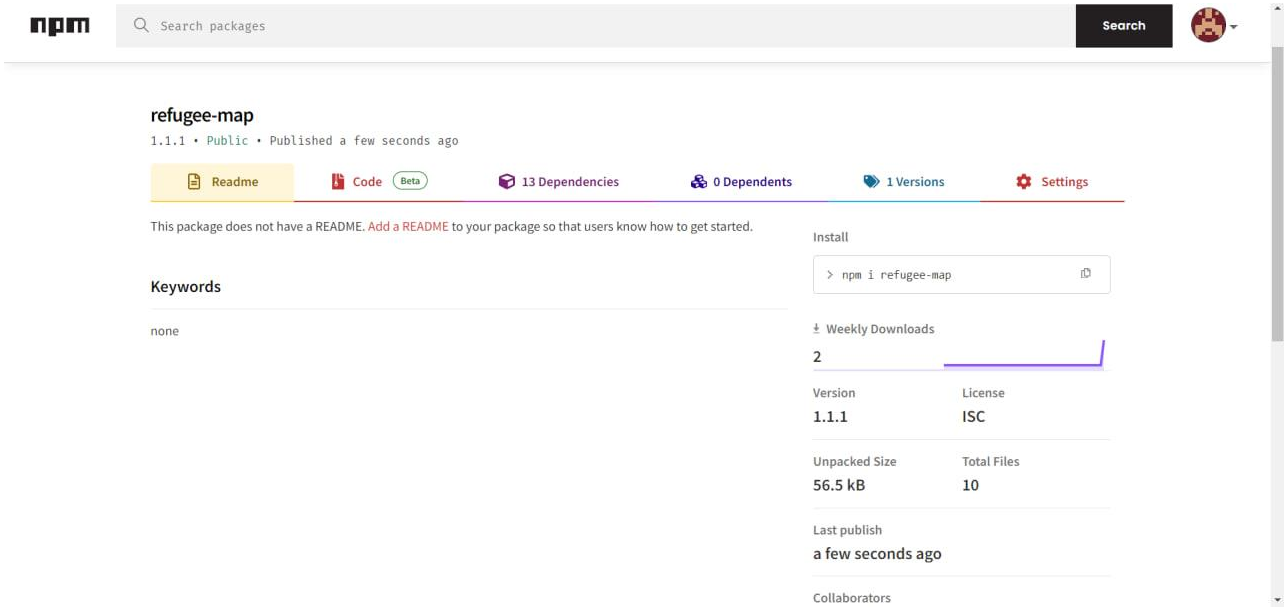

Рисунок 3.1 – Процес публікації модуля на npm для інтеграції в інші проекти

Цей процес демонструє, як можна розробити, інтегрувати та публікувати власні модулі на npm, що є важливою частиною розробки JavaScript. Інші розробники можуть використовувати ці модулі у своїх власних проектах, що значно спрощує процес розробки.

3.5 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки програмних модулів. Виконавши аналіз програмного продукту та його задач було обрано React, node.js, Mongoose та Express.js. Для зручності розробки графічного інтерфейсу користувача було обрано застосунок WebStorm. Для роботи із інтерактивної картою було використано react-leaflet. Також для реалізації кластеризації було використано бібліотеку use-superclasster для групування маркерів задля полегшення навігації користувачів та зменшення навантаження на застосунок. У розділі також було описано розробку основних програмних модулів кластеризації маркерів, та відображення маркерів на інтерактивній карті. Окрім цього, акцентується увага на інтеграції цих модулів в єдиний модуль та їх подальшої публікації на npm.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Тестування програмного забезпечення - [22] це процес технічного аналізу, спрямований на виявлення інформації про якість продукту у відношенні до контексту його використання. Ця техніка включає як процес пошуку помилок або інших дефектів, так і випробування програмних компонентів для оцінки їхньої ефективності.

Тестування грає важливу роль у розробці програмного забезпечення з декількох причин. По-перше, воно дозволяє виявити помилки та недоліки, які можуть призвести до некоректної роботи програми або порушити безпеку системи, забезпечуючи високу якість продукту та задоволення користувачів. По-друге, тестування допомагає перевірити, чи відповідає програма вимогам та очікуванням, визначеним у специфікації.

Перший етап - це функціональне тестування [23], спрямоване на перевірку відповідності функціональних вимог програмного забезпечення його реальним характеристикам. Основним завданням цього етапу є підтвердження того, що розроблений програмний продукт володіє усім необхідним функціоналом для замовника.

Другий етап - це тестування відмов [24], головною метою якого є визначення здатності програмного забезпечення до опору та відновлення після виникнення збоїв у його роботі, незалежно від того, чи вони виникли всередині програми, чи за її межами.

Після завершення функціонального та тестування відмов, проводиться додаткове тестування для визначення продуктивності програмного забезпечення. Цей етап включає тести швидкодії, спрямовані на оцінку швидкості виконання певних операцій або завдань програмою. Під час цього тестування вимірюється час відповіді програми на різноманітні запити або дії користувача.

Окремим етапом тестування може бути тестування сумісності [25], яке призначене для визначення ступеня сумісності програмного забезпечення з певним середовищем — операційною системою, платформою чи обладнанням.

Першим кроком тестування розробленого програмного забезпечення буде тестування для перевірки роботи системи реєстрації. У разі спроби ввести різні паролі в поля «Пароль» та «Підтвердити пароль» застосунок повідомить користувачу про те що паролі відрізняються (рис 4.1).

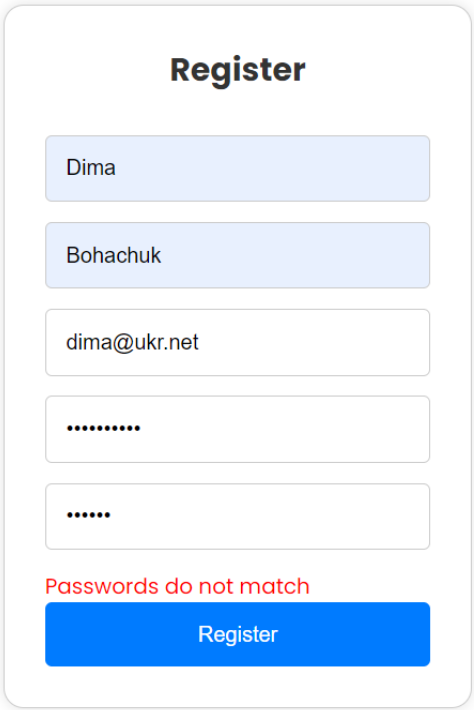

Рисунок 4.1 – Введеня різних паролів

Для забезпечення надійності та коректності роботи системи реєстрації та авторизації, було вирішено провести додаткове тестування з введенням пустих даних. Це допоможе переконатися, що система правильно обробляє такі ситуації та виводить відповідні повідомлення про помилку, що допоможе користувачам правильно заповнити форми реєстрації та авторизації. Також це допоможе переконатися, що користувач не може зареєструватися або увійти в систему без введення необхідних даних. Реакція системи наведена на рисунку 4.2.

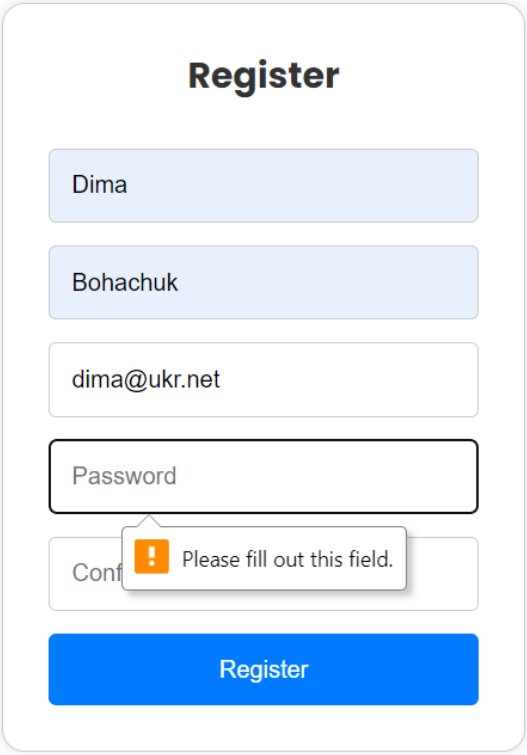

Рисунок 4.2 – Спроба реєстрації без вхідних даних

Наступним кроком буде перевірка програмного забезпечення на систему авторизації, для перевірки введемо email та не правильний пароль існуючого аккаунта й перевіримо чи виконається авторизація. Реакція системи наведена на рисунку 4.3.

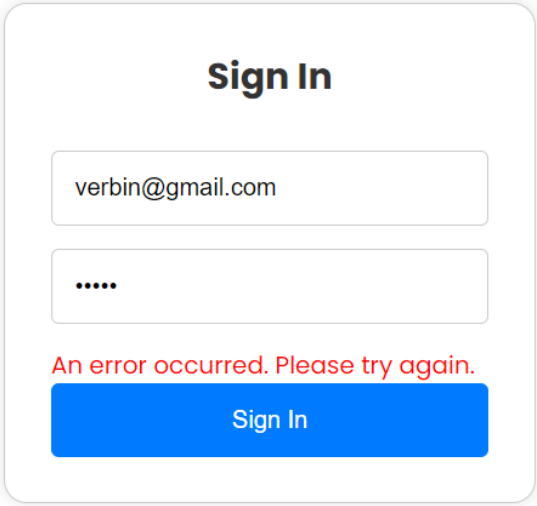

Рисунок 4.3 – Спроба авторизації без введення коректних даних

Для забезпечення коректної роботи системи та відповідності її вимогам користувачів, важливо перевірити функціональність профілів користувачів з різними ролями. В нашому випадку, ми маємо дві основні ролі: біженець та волонтер.

Біженці - це основні користувачі нашого додатку, які шукають допомогу. Вони можуть переглядати доступні пропозиції допомоги, фільтрувати їх за різними параметрами та вибирати найбільш підходящі для себе. Тому ми проведемо тестування, щоб переконатися, що вони можуть без проблем виконувати ці дії, увійдемо до профілю користувача з роллю біженця (рис. 4.4).

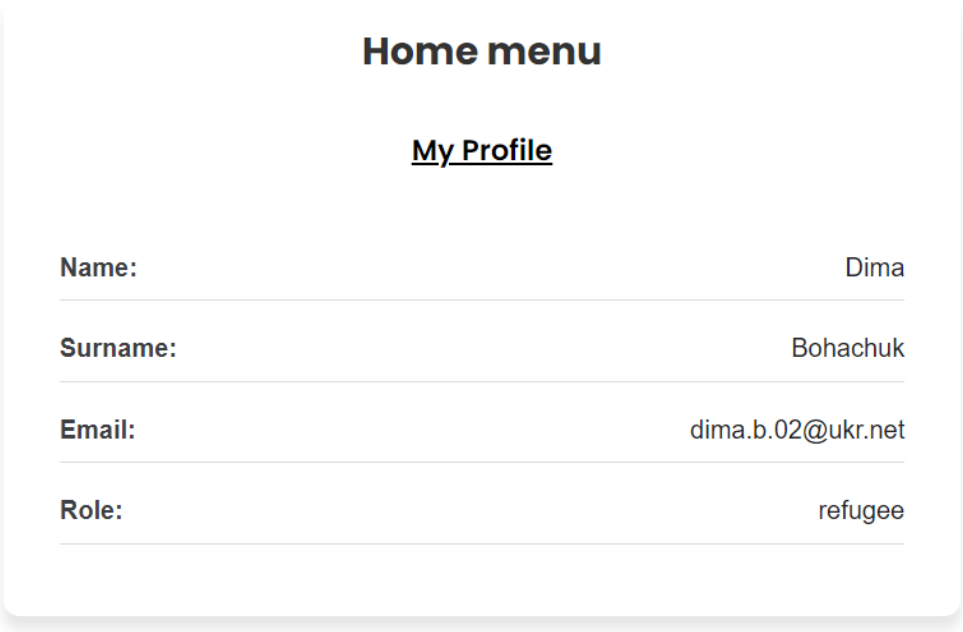

Рисунок 4.4 – Відображення аккаунта з роллю біженця

Волонтери - це користувачі, які надають допомогу. Вони можуть переглядати свій власний профіль, створювати нові пропозиції допомоги, або видаляти свої існуючі пропозиції. Ми проведемо тестування, щоб переконатися, що вони можуть без проблем виконувати ці дії. Для початку спробуємо відкрити профіль користувача з роллю волонтер (рис. 4.5).

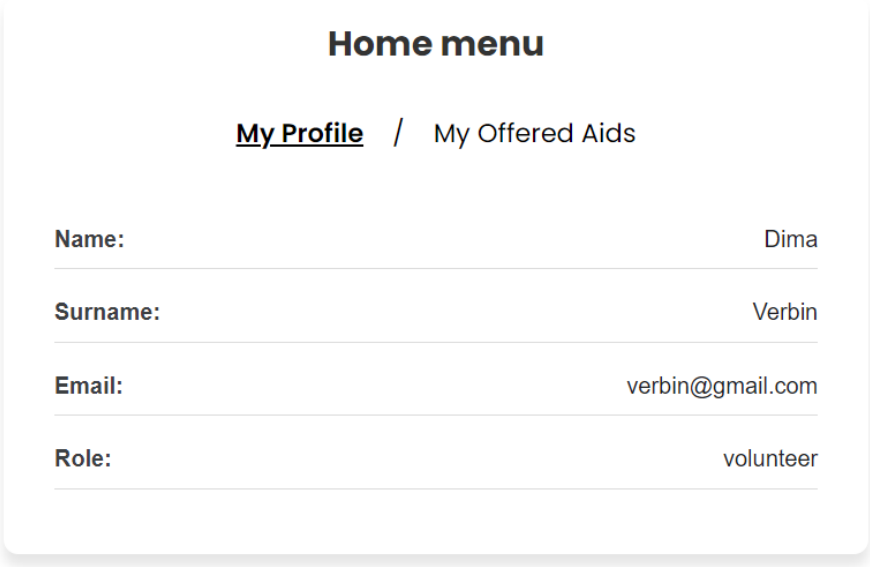

Рисунок 4.5 – Відображення аккаунта з роллю волонтера

Після успішного входу до профілю волонтера почнемо з перевірки можливості волонтерів створювати нові пропозиції допомоги, а також перевіримо, чи відображається нова пропозиція в списку пропозицій допомоги волонтера. Для тестування зробимо перевірку форми додадання нової пропозиції, додавши допомогу на карту без введених в неї даних (рис. 4.6).

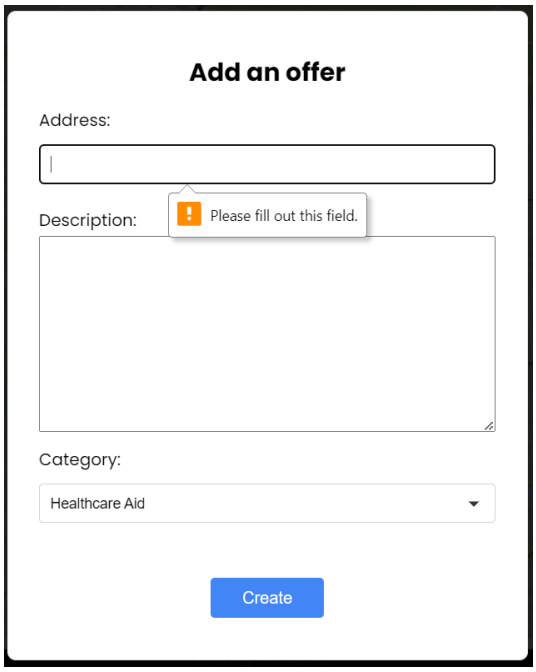

Рисунок 4.6 – Спроба додати пропозицію допомоги без введеня інформації

Спробуємо додати пропозицію допомоги вводячи дані у формі, для перевірки правильної роботи нашого застосунку (рис. 4.7).

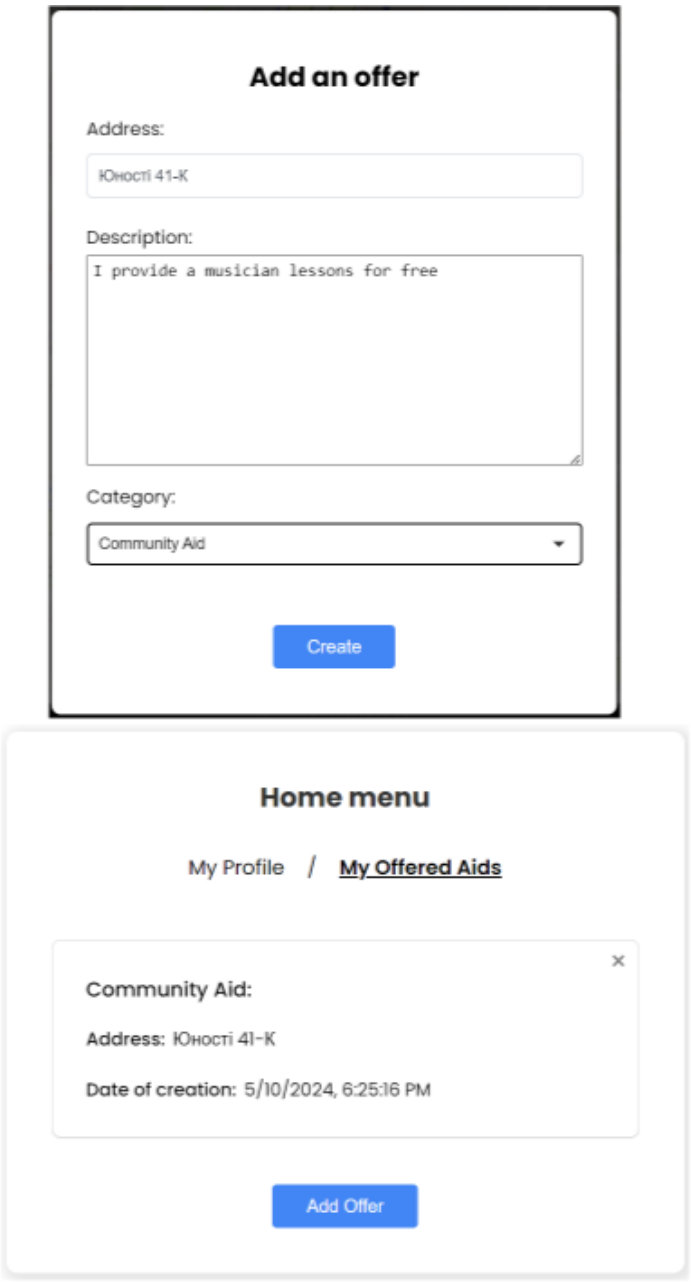

Рисунок 4.7 – Результат успішного створення пропозиції допомоги

Після успішного створення нової пропозиції, ми перевіримо, чи можуть волонтери видаляти свої існуючі пропозиції допомоги. Ми спробуємо видалити одину з існуючих пропозицій та перевіримо, чи видаляється вона з історії волонтера (рис. 4.8).

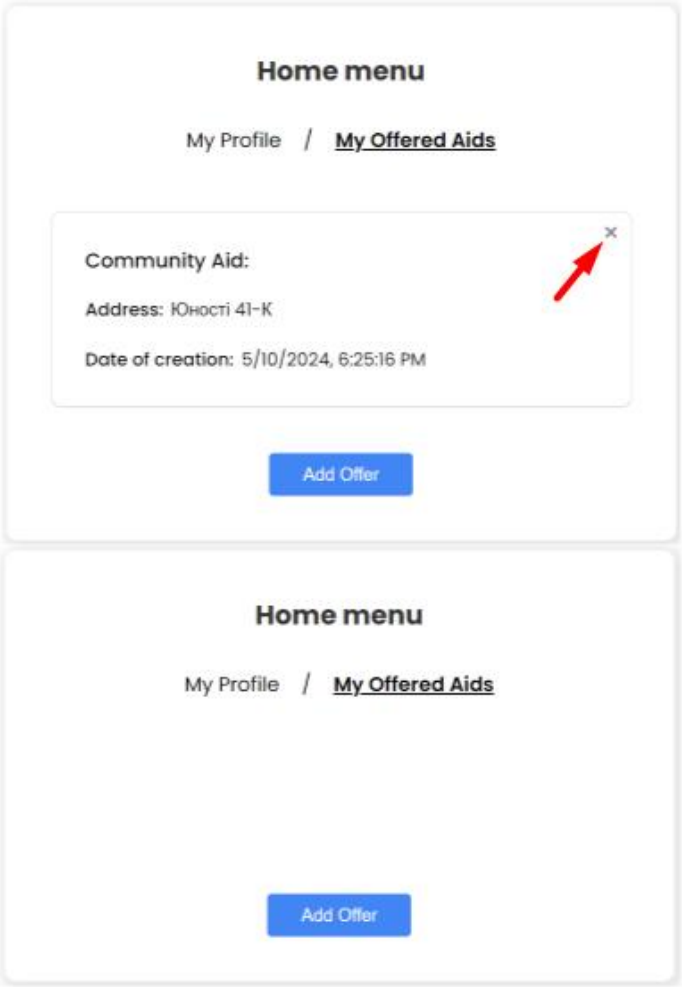

Рисунок 4.8 – Успіше видалення існуючої пропозиції

Після успішного тестування основних функцій системи, наступним кроком було тестування розроблених модулів, які були опубліковані на npm.

Спочатку створемо новий react проект через npx create-react-app та виконаємо команду npm install refugee-map в терміналі нашого проекту. Це дозволить нам встановити модуль refugee-map, щоб мати можливість користуватися розробленими модулями (рис. 4.9).

```
PS D:\React project\fashion-react> npm install refugee-map

up to date, audited 1611 packages in 6s
```

Рисунок 4.9 – Встановлення модуля refugee-map

Після успішного встановлення модулів, їх можна імпортувати в проект використовуючи команду import Map from 'refugee-map'. Це дозволить використовувати розроблені модулі для відображення маркерів на інтерактивній карті та кластеризації маркерів в новому проєкті.

Після успішного імпорту модулів у наш проект, ми використали їх для перевірки відображення маркерів у різних браузерах. Тестування відбулося в Google Chrome, оскільки цей браузер є найбільш популярним серед користувачів. Результати використання модулю в Google Chrome показали успішне відображення маркерів (рис. 4.10).

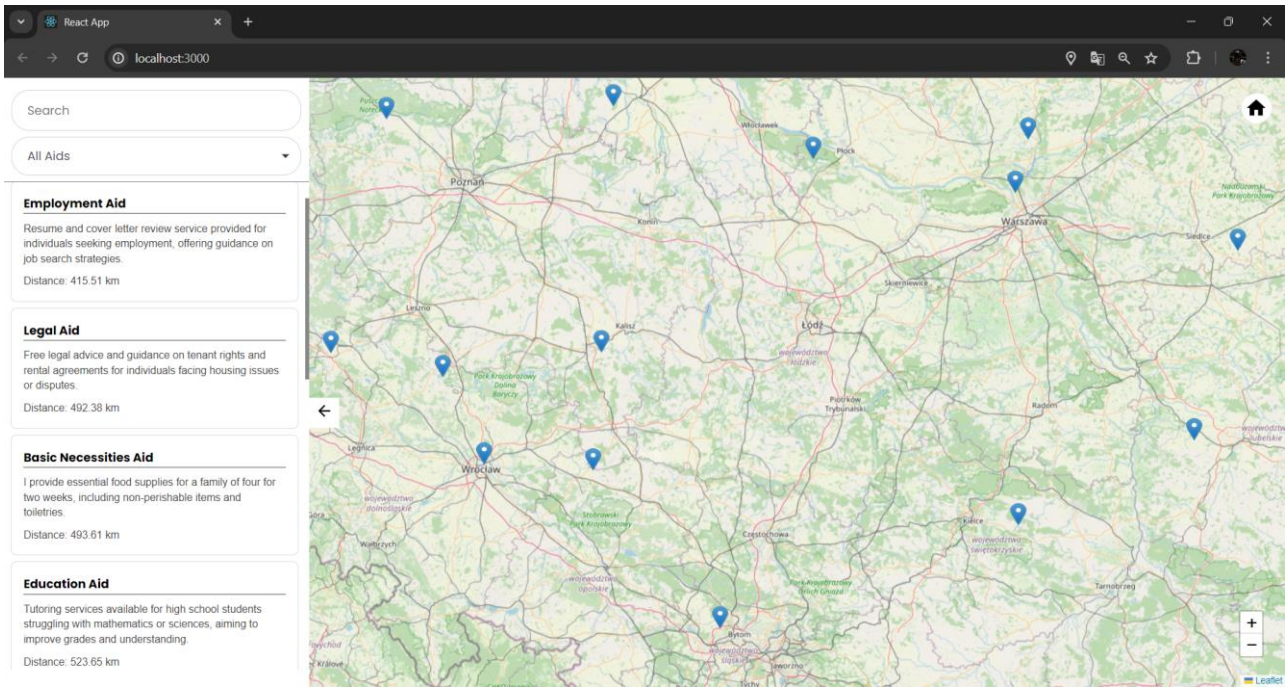

Рисунок 4.10 – Сумісність нашого модулю з браузером Google Chrome

Після цього ми провели перевірку кластеризації маркерів у Microsoft Edge, браузері, що посідає друге місце у світовому рейтингу популярності, відступаючи лише від Google Chrome. Переконавшись у коректності відображення кластерів, ми отримали позитивні результати які відображаються на рисунку 4.11.

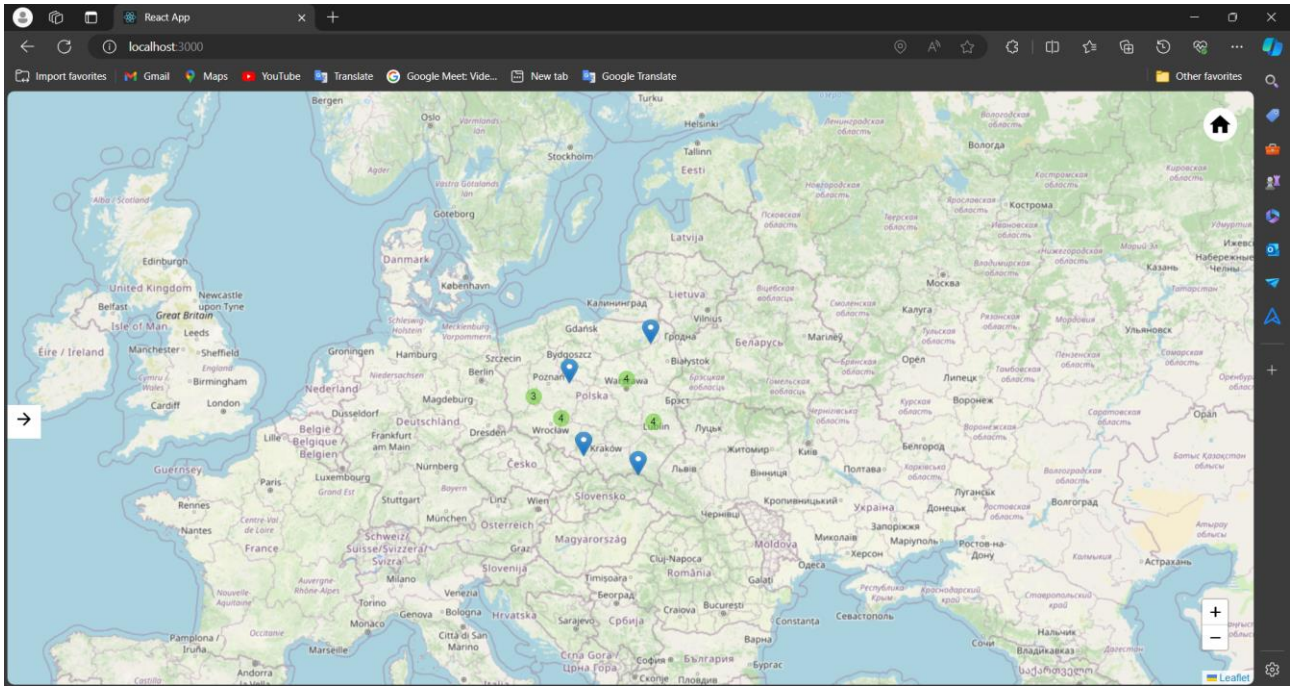

Рисунок 4.11 – Сумісність нашого модулю з браузером Microsoft Edge

Останнім етапом тестування розробленого програмного застосунку було перевірка швидкості відгуку системи на дії користувача. Під час тестування було встановлено, що час реакції системи на взаємодію користувача з інтерфейсом не перевищує 0.5 секунди.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації розміщено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	**Рекомендовано**	**Мінімально**
Процесор	Будь-який, що підтримує сучасні версії веб-браузерів	Будь-який, що підтримує сучасні версії веб-браузерів
Оперативна пам'ять	4 ГБ RAM	2 ГБ RAM

Продовження таблиці 4.1

Вільний простір на диску	500 МБ	200 МБ
Відеокарта	Будь-яка, що забезпечує базове відображення веб-сайтів	Будь-яка, що забезпечує базове відображення веб-сайтів
Операційна система	Будь-яка сучасна версія Windows, macOS, Linux, android або IOS	Будь-яка сучасна версія Windows, macOS, Linux, android або IOS
Монітор	Роздільна здатність не менше 1280x720 пікселів, діагональ 15" або більше	Роздільна здатність не менше 1024x768 пікселів, діагональ 13" або більше

Відкриваючи портал у звичайному веб-браузері, користувачі без жодних додаткових завантажень можуть отримати доступ до усіх його функцій.

Головна сторінка вітає гостей лентою новин, яка постійно оновлюється, щоб надати актуальну інформацію та останні оновлення (рис. 4.12). Крім того, користувачі можуть взаємодіяти з інтерактивною картою, яка надає додаткову інформацію.

Для того щоб стати повноцінним користувачем порталу, гостям варто створити аккаунт. Тим, хто вже має свій обліковий запис, просто необхідно авторизуватися. Після цього вони зможуть скористатися всіма можливостями порталу.

Користувачі з роллю біженця отримують доступ до свого профілю, де можуть переглядати інформацію, що стосується лише їх самих.

Користувачі з роллю волонтера мають більш широкі можливості. Вони можуть переглядати свою власну історію створених пропозицій допомоги, а також створювати нові пропозиції та видаляти існуючі.

Така структура дозволяє кожному користувачеві максимально зручно користуватися порталом, забезпечуючи необхідну інформацію та можливості для кожного типу користувача.

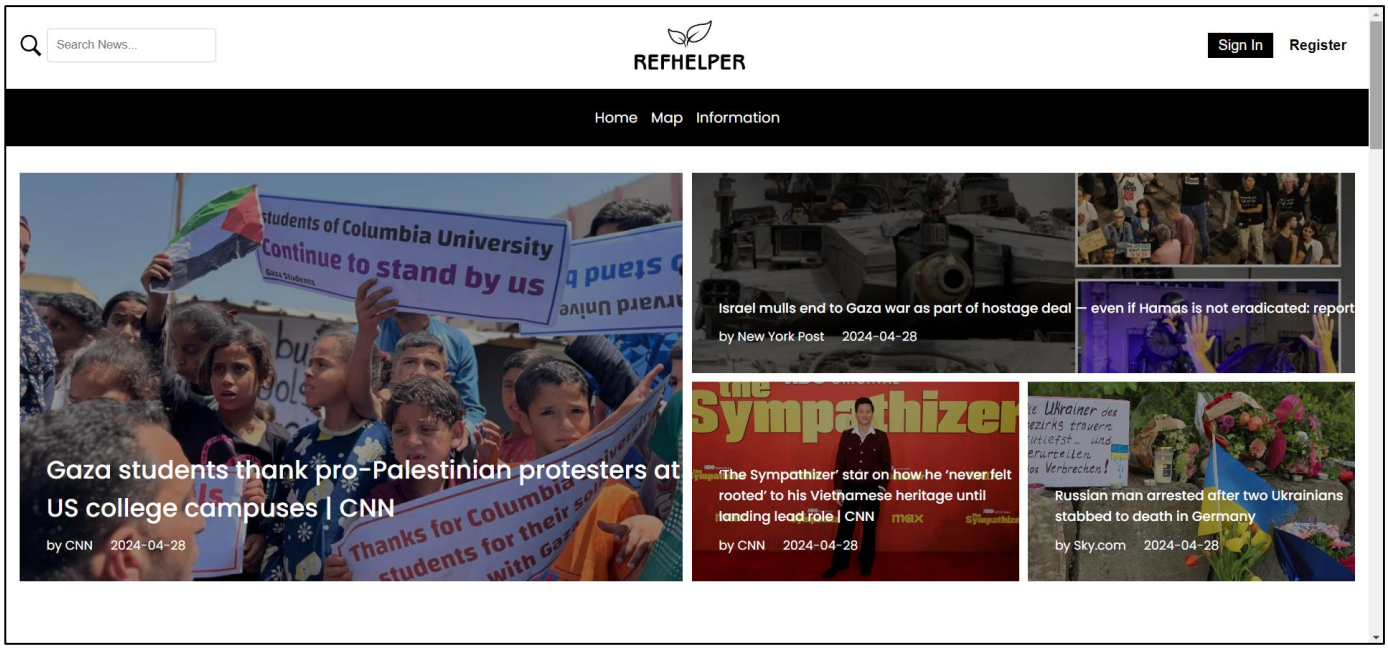

Рисунок 4.12 – Головна сторінка онлайн-порталу для біженців

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями. Було перевірено реакцію програмних засобів на помилкові ситуації та вхідні дані різних форматів. Також було розроблено інструкцію користувача по початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській дипломній роботі були розроблені модулі програмного забезпечення для інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями. Для розробки використовувалося середовище програмування WebStorm.

Під час виконання бакалаврської роботи було досліджено поточний стан проблеми. Були розглянуті основні аналоги програмного продукту, виявлені їхні недоліки та порівняно з власним розробленим продуктом. На основі цього порівняння були сформульовані основні завдання.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування JavaScript, середовища node.js, фреймворку Express.js, а також бібліотек React, use-superclaster, Leaflet та Mongoose.

У бакалаврській роботі було зроблено наступне:

– розроблено модуль для кластеризації маркерів на мапі, що дозволить групувати подібні об'єкти для зручності навігації користувачів;

– розроблено алгоритм для збереження історії маркерів, для перегляду попередніх запитів та дій на мапі;

– розроблено власну карту волонтера для можливості створення нової допомоги;

– розроблено систему авторизації та реєстрації користувачів з використанням сучасних методів шифрування даних;

– розроблено розділ з новинами для відслідковування та відображення актуальних новин та постів у світі;

– розроблено модуль графічного інтерфейсу для відображення маркерів на мапі з інтерактивними можливостями взаємодії для користувачів;

– здійснено інтеграцію різних модулів в єдиний програмний засіб для забезпечення функціональності порталу та його використання користувачами;

– проведено тестування програмного застосунка згідно поставлених задач.

Було розроблено схеми загального алгоритму роботи програмного застосунку, модуля відображення маркерів на інтерактивній карті та модуля кластеризації маркерів. Також було розроблено модель системи з використанням UML діаграм.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

ЛІТЕРАТУРА

1. Богачук Д. В., Чехместрук Р. Ю. Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

2. Online Refugee Aid Portal. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tjugw.

3. The Refugee Project. [Електронний ресурс] – Режим доступу до ресурсу: https://www.therefugeeproject.org/.

4. Refugee Integration Network. [Електронний ресурс] – Режим доступу до ресурсу: https://www.dcu.ie/irin/ua.

5. RefAid. [Електронний ресурс] – Режим доступу до ресурсу: https://refaid.com/.

6. Scottish Refugee Council Integration Network. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tjuio.

7. LucidChart. [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/landing?.

8. JavaScript. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript.

9. Павленко В. С. Angular: повний посібник для розробників. Київ: Наука і Техніка, 2022. 456 с.

10. Ковальчук І. В. Vue.js: від новачка до професіонала. Львів: Техно Прогрес, 2023. 398 с.

11. Шевченко О. Л. React JS: глибокий погляд на розробку. Харків: Інновації, 2021. 520 с.

12. VSCode [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com.

13. WebStorm. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tjuns.

14. Atom [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tqidg.

15. Leaflet. [Електронний ресурс] – Режим доступу до ресурсу: https://leafletjs.com/.

16. Mongoose. [Електронний ресурс] – Режим доступу до ресурсу: https://code.tutsplus.com/uk/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527a.

17. MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikiedia.org/wiki/MongoDB.

18. Node.js. [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/.

19. Express.js. [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/.

20. Формула гаверсинусів. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tjujl.

21. Кластеризація. [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/tjujt.

22. Ткаченко П. І. Тестування програмного забезпечення: методи та інструменти. Київ: ІТ Академія, 2020. 412 с.

23. Бондаренко Н. А. Функціональне тестування: теорія і практика. Одеса: Південь, 2021. 385 с.

24. Петренко О. Ю. Тестування відмов: новітні підходи та методи. Вінниця: Видавничий Дім, 2023. 290 с.

25. Іванова Л. М. Тестування сумісності: посібник для тестувальників. Дніпро: Сучасні Технології, 2022. 305 с.

ДОДАТКИ

61

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. каф. ПЗ Чехместрук Р.Ю.
«12» березня 2024 р.

Виконав:
_____ студент гр. 2ПІ-20б Д. В. Богачук
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями».

Галузь застосування – соціальні послуги.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою дослідження є підвищення доступу до інформації про допомогу біженцям яке являється важливим кроком у забезпеченні гуманітарної допомоги та соціальної інтеграції для тих, хто опинився у складних життєвих обставинах через міграцію.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Богачук Д. В., Чехместрук Р. Ю. Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями // Стан, досягнення та перспективи інформаційних систем і технологій: матеріали XXIV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів, Одеса, 18-19 квітня 2024 р. Одеса: Видавництво ОНТУ, 2024. 498 с.

5. Технічні вимоги

Вихідні дані до роботи: середовище розробки WebStorm, мова розробки JavaScript, бібліотека React, протокол передачі даних HTTP, системи управління базою даних MongoDB, алгоритм кластеризації.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ з\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану наявних онлайн-порталів для біженців	13.03.24 – 31.03.24
2	Розробка інтерфейсу, моделі та алгоритмів програмного продукту	01.04.24 –20.04.24
3	Вибір середовища та мови розробки	21.04.24 – 04.05.24
4	Розробка модуля кластеризації маркерів	05.05.24 – 14.05.24
5	Розробка модуля для відображення маркерів на інтерактивній карті	15.05.24 – 19.05.24
6	Тестування	20.05.24 – 29.05.24
7	Оформлення матеріалів до захисту БДР	30.05.24 – 31.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

64

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інтегрованого онлайн-порталу для біженців з персоналізованими рекомендаціями

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Чехместрук Р.Ю.

Оригінальність	90,3
Схожість	9,7

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Богачук Д.В.

Керівник роботи Чехместрук Р.Ю.

Додаток В – Лістинг програми

//index.js

```
import React from "react"
import ReactDOM from "react-dom/client"
import App from "./App"
import { QueryClient, QueryClientProvider } from 'react-query';


const client = new QueryClient();
const root = ReactDOM.createRoot(document.getElementById("root"))
root.render(
<QueryClientProvider client={client}>
   <App />
</QueryClientProvider>
)
```

// компонент app.js

```
import React from 'react';
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom';
import Header from './components/common/header/Header';
import SignInForm from './components/auth/signInForm/SignInForm';
import RegistrationForm from './components/auth/registrationForm/RegistrationForm';
import {AuthProvider} from "./components/AuthContext/AuthProvider";
import {NewsProvider}  from './components/newsProvider/NewsProvider';
import Map from "./components/map/Map";
import SinglePage from "./components/singlePage/SinglePage";
import Homepages from "./components/home/Homepages";
import Footer from "./components/common/footer/Footer";
import Profile from "./components/profile/Profile";
import "./App.css"


const App = () => {
   return (
      <AuthProvider>
         <NewsProvider>
            <Router>
               <Switch>
                  <Route path="/authorisation" component={SignInForm} />
                  <Route path="/register" component={RegistrationForm} />
                  <Route>
                     <Header />
                     <Switch>
                        <Route exact path='/' component={Homepages} />
                        <Route path="/profile" component={Profile} />
                        <Route path='/singlepage/:keyword/:id' exact component={SinglePage} />
                        <Route exact path='/map' component={Map} />
                     </Switch>
                     <Footer />
                  </Route>
               </Switch>
            </Router>
         </NewsProvider>
      </AuthProvider>
   );
```

```
};

export default App;
```

// компонент NewsProvider

```
import React, {createContext, useContext, useEffect, useState} from "react";
import NewsService from "../../services/NewsService";

const NewsContext = createContext();

export const useNewsContext = () => useContext(NewsContext);

export const NewsProvider = ({ children }) => {
   const [categoriesData, setCategoriesData] = useState({});
   const { getAllNews, getRequestValues } = NewsService();

   useEffect(() => {
      const updateData = async () => {
         const updatedCategoriesData = {};
         try {
            for (const category in getRequestValues) {
               const data = await getAllNews(category);
               updatedCategoriesData[category] = data;
            }
            return updatedCategoriesData;
         } catch (error) {
            console.error("Error updating data:", error);
         }
      };

      updateData().then(res => {
         if (res) {
            setCategoriesData(res);
         }
      });

      const interval = setInterval(updateData, 12 * 60 * 60 * 1000);

      return () => clearInterval(interval);
   }, []);

   return (
      <NewsContext.Provider value={{ categoriesData }}>
         {Object.keys(categoriesData).length > 0 && children}
      </NewsContext.Provider>
   );
};
```

//Authprovider

```
import React, {createContext, useEffect, useState} from 'react';

export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
   const [user, setUser] = useState(null);
```

```
  useEffect(() => {
    const token = localStorage.getItem('token');
    const email = localStorage.getItem('email');
    if (token && email) {
      setUser({ email, token });
    }
  }, []);

  const login = (userData) => {
    localStorage.setItem('token', userData.token);
    localStorage.setItem('email', userData.email);
    setUser(userData);
  };

  const logout = () => {
    localStorage.removeItem('token');
    localStorage.removeItem('email');
    setUser(null);
  };

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};
```

// Компонент авторизації

```
import React, { useState, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from '../../AuthContext/AuthProvider';
import { useHistory } from 'react-router-dom';
import '../auth.css'

const SignInForm = () => {
  const { login } = useContext(AuthContext);
  const history = useHistory();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://localhost:3001/auth/login', {
        email,
        password
      });
      if (response.data.token) {
        login({ email, token: response.data.token })
        history.push('/');
      } else {
        setError('Invalid email or password');
      }
    } catch (error) {
      setError('An error occurred. Please try again.');
    }
```

```
  };

  return (
    <div className="auth-form">
      <h2>Sign In</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="email"
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <input
          type="password"
          placeholder="Password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
        {error && <p>{error}</p>}
        <button type="submit">Sign In</button>
      </form>
    </div>
  );
};

export default SignInForm;
```

// Компонент реєстрації

```
// RegistrationForm.js
import React, { useState, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from '../../AuthContext/AuthProvider';
import { useHistory } from 'react-router-dom';
import '../auth.css';

const RegistrationForm = () => {
  const history = useHistory();
  const { login } = useContext(AuthContext);
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (password !== confirmPassword) {
      setError('Passwords do not match');
      return;
    }
    try {
      const data =  {firstName,
        lastName,
        email,
        password}
```

```
            const response = await axios.post('http://localhost:3001/auth/registration', data);
            if (response.data.message === 'Користувач успішно зареєстрований') {
                setError('');
                history.push('/');
            } else {
                setError('Registration failed. Please try again.');
            }
        } catch (error) {
            setError('An error occurred. Please try again.');
        }
    };

    return (
        <div className="auth-form">
            <h2>Register</h2>
            <form onSubmit={handleSubmit}> <input
                type="text"
                placeholder="First Name"
                value={firstName}
                onChange={(e) => setFirstName(e.target.value)}
                required
            />
                <input
                    type="text"
                    placeholder="Last Name"
                    value={lastName}
                    onChange={(e) => setLastName(e.target.value)}
                    required
                />
                <input
                    type="email"
                    placeholder="Email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                    required
                />
                <input
                    type="password"
                    placeholder="Password"
                    value={password}
                    onChange={(e) => setPassword(e.target.value)}
                    required
                />
                <input
                    type="password"
                    placeholder="Confirm Password"
                    value={confirmPassword}
                    onChange={(e) => setConfirmPassword(e.target.value)}
                    required
                />
                {error && <p style={{color: 'red'}}>{error}</p>}
                <button type="submit">Register</button>
            </form>
        </div>
    );
};

export default RegistrationForm;
```

//Меню навігації

```
import React, { useState } from "react"
import Head from "./Head"
import "./header.css"
import { Link } from "react-router-dom"

const Header = () => {
  const [navbar, setNavbar] = useState(false)

  return (
    <>
      <Head />
      <header>
        <div className='container paddingSmall'>
          <nav>
            <ul className={navbar ? "navbar" : "flex"} onClick={() => setNavbar(false)}>
              <li>
                <Link to='/'>Home</Link>
              </li>
              <li>
                <Link to='/map'>Map</Link>
              </li>
              <li>
                <Link to='/information'>Information</Link>
              </li>
            </ul>
            <button className='barIcon' onClick={() => setNavbar(!navbar)}>
              {navbar ? <i className='fa fa-times'></i> : <i className='fa fa-bars'></i>}
            </button>
          </nav>
        </div>
      </header>
    </>
  )
}



export default Header
```

//Шапка порталу

```
import React, {useContext, useState} from "react"
import {Link} from "react-router-dom";
import {AuthContext} from "../../AuthContext/AuthProvider";

const Head = () => {
    const { user, logout } = useContext(AuthContext);
    const [searchText, setSearchText] = useState('');

    const handleLogout = () => {
        logout(null);
    };

    return (
        <>
            <section className='head'>
                <div className='container flexSB'>
                    <div className='search'>
```

```
                <svg viewBox="0 0 32 32" width="30" height="30" category="actions" icon="search" className="sc-
2c06e71a-0 fsMljb"><path d="m30.6 28.1-8.3-8.3c1.5-2 2.4-4.4 2.4-7.2C24.7 6 19.6 1 13 1S1.4 6.1 1.4 12.7 6.5 24.3
13 24.3c2.3 0 4.4-.6 6.2-1.8l8.5 8.5 2.9-2.9zM4 12.6c0-5.2 3.9-9.1 9-9.1s9 3.9 9 9.1c0 5.2-3.9 9.1-9 9.1s-9-3.9-9-
9.1z"></path></svg>
                <input
                  value={searchText}
                  className="search-input"
                  type="text"
                  placeholder="Search News..."
                  onChange={(e) => setSearchText(e.target.value)}
                />
              </div>
              <div className='logo'>
                <Link to="/">
                  <img src='../images/Logo.png' alt='' />
                </Link>
              </div>
              <div className='auth-buttons'>
                {user ? (
                  <>
                    <Link to={'/profile'}>{user.email}</Link>
                    <button className="auth-button" onClick={handleLogout}>Logout</button>
                  </>
                ) : (
                  <>
                    <button className="auth-button"><Link to="/authorisation">Sign In</Link></button>
                    <button className="auth-button"><Link to="/register">Register</Link></button>
                  </>
                )}
              </div>
            </div>
          </section>
        </>
    )
}


export default Head
```

//Компонент з головними новинами

```
import React, {useEffect, useMemo, useState} from "react"
import {useNewsContext} from "../../newsProvider/NewsProvider";
import "./hero.css"
import Card from "./Card"

const Hero = () => {
    const {categoriesData } = useNewsContext();

    const filteredItems = useMemo(() => categoriesData.usNews.slice(0, 4), [categoriesData.usNews]);

    return (
        <>{
            filteredItems ? (
                <section className='hero'>
                    <div className='container'>

                        {filteredItems.map((item, i) => (
                            <Card key={item.id || i} item={item} />
```

```
                ))}
              </div>
            </section>
          )
            : <p>Loading...</p>
      }

      </>
    )
}

export default Hero
```

//Компонент новини

```
import React from "react"
import { Link } from "react-router-dom"

const Card = ({ item: { id, cover, title, name, time, keyword } }) => {
  return (
    <>
      <Link to={`/SinglePage/${keyword}/${id}`} className='box'>
        <div className='img'>
          <img src={cover} alt='' />
        </div>
        <div className='text'>
          <h1 className='titleBg'>{title}</h1>
          <div className='author flex'>
            <span>by {name}</span>
            <span>{time}</span>
          </div>
        </div>
      </Link>
    </>
  )
}

export default Card
```

// Інші новини порталу

```
import React from "react"
import Side from "../../sideContent/side/Side"
import Life from "../life/Life"
import Music from "../musics/Music"
import Popular from "../popular/Popular"
import Ppost from "../Ppost/Ppost"
import "./style.css"

const Homes = () => {
  return (
    <>
      <main>
        <div className='container'>
          <section className='mainContent'>
            <Popular />
            <Ppost />
            <Life />
```

```
          <Music />
        </section>
        <section className='sideContent'>
          <Side />
        </section>
      </div>
    </main>
  </>
 )
}

export default Homes
```

//Компонент для відкриття обраної новини

```
import React, {useEffect, useMemo, useState} from "react"
import { useParams } from "react-router-dom"
import Side from "../home/sideContent/side/Side"
import "../home/mainContent/homes/style.css"
import "./singlepage.css"
import "../home/sideContent/side/side.css"
import SinglePageSlider from "./slider/SinglePageSlider"
import {useNewsContext} from "../newsProvider/NewsProvider";

const SinglePage = () => {
  const { id, keyword } = useParams();
  const [item, setItem] = useState(null);
  const {categoriesData} = useNewsContext();


   useEffect(() => {
      if (categoriesData && categoriesData[keyword]) {
         const filteredItem = categoriesData[keyword].find((item) => item.id === parseInt(id));
         setItem(filteredItem);
         window.scrollTo(0, 0);
      }
   }, [categoriesData, id, keyword]);


  return (
   <>
      {item ? (
         <main>
            <SinglePageSlider object={categoriesData[keyword]}/>
            <div className='container'>
               <section className='mainContent details'>
                  <h1 className='title'>{item.title}</h1>

                  <div className='author'>
                     <span style={{marginRight: 5}}>by </span>
                     <p> {item.authorName} on</p>
                     <label>{item.time}</label>
                  </div>

                  <div className='social'>
                     <div className='socBox'>
                        <i className='fab fa-facebook-f'></i>
                        <span>SHARE</span>
                     </div>
                     <div className='socBox'>
```

```
                        <i className='fab fa-twitter'></i>
                        <span>TWITTER</span>
                      </div>
                      <div style={{width: 55}} className='socBox'>
                        <i className='fab fa-pinterest'></i>
                      </div>
                      <div style={{width: 55}} className='socBox'>
                        <i className='fa fa-envelope'></i>
                      </div>
                    </div>

                    <div className='desctop'>
                      <p>{item.description}</p>
                    </div>
                    <img style={{maxWidth: '100%', marginTop: 0}} src={item.cover} alt='' />


                    <div className='descbot'>
                      <p>{item.content}</p>
                    </div>

                    <div style={{marginTop: 40, marginBottom: 40}} className='quote'>
                      <i className='fa fa-quote-left'></i>
                      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elita  sunt suscipit tempora tempore ullam vel, velit veniam veritatis vero voluptas voluptates voluptatum? enim harum inventore iure libero nobis reiciendis saepe, tempora veritatis voluptate. Ex fugit quae quasi unde vitae?</p>
                    </div>

                    <div className='desctop'>
                      <p>Nobody consectetur adipisicing elit. Consequatur distinctio doloribus eius, fuga illo illum laboriosam laudantium, nam nobis odio perspiciatis porro quae quam quod sapiente velit veritatis. Beatae cupiditate, distinctio doloribus eveniet expedita, incidunt ipsa iure nam praesentium quas quis quisquam repudiandae rerum sequi sunt, veniam voluptas. Aliquid ex expedita natus, perspiciatis quidem quod totam ut. Asperiores cupiditate dolorem doloribus excepturi inventore itaque natus nihil quisquam, igendi, esse eum facere hic, impedit ipsum nostrum quo quod ratione sit.</p>
                      <p>Everything dolor sit amet, consectetur adipisicing elit. Consequatur distinctio doloribus eius, fuga illo illum laboriosam laudantium, nam nobis odio perspiciatis porro quae quam quod sapiente velit veritatis. Beatae cupiditate, distinctio doloribus eveniet expedita, incidunt ipsa iure nam praesentium quas quis quisquam repudiandae rerum sequi sunt, veniam voluptas. Aliquid ex expedita natus, perspiciatis quidem quod totam ut. Asperiores cupiditate dolorem doloribus excepturi inventore itaque natus nihil quisquam, .</p>
                      <p>Beatae cupiditate, distinctio doloribus eveniet expedita, incidunt ipsa iure nam praesentium quas quis quisquam repudiandae rerum sequi sunt, veniam voluptas. Aliquid ex expedita natus, perspiciatis quidem quod totam ut. Asperiores cupiditate dolorem doloribus excepturi inventore itaque natus nihil quisquam, sapiente veniam! Aspernatur atque beatae eaque facilis laborum laudantium necessitatibus sed sit sunt voluptate. Consequatur expedita impedit velit. Accusantium deserunt distinctio, eligendi, esse eum facere hic, impedit ipsum nostrum quo quod ratione sit.</p>
                      <p>Sit amet, consectetur adipisicing elit. Aliquid eos et eum hic minima nihil non odit quas ullam ut! Ipsa ipsam minus omnis perferendis quas, quasi saepe sapiente.</p>
                    </div>
                  </section>
                  <section className='sideContent'>
                    <Side />
                  </section>
                </div>
              </main>
            ) : null}

        </>
      )
    }
```

```
export default SinglePage
```

// Компонент карти

```
import React, { useEffect, useState } from 'react';
import { useQuery } from "react-query";
import { MapContainer, TileLayer } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';
import ShowCrimes from "./ShowCrimes";
import './map.css';
import L from 'leaflet';

const fetchMarkers = async () => {
  const response = await fetch('http://localhost:3001/api/markers');
  if (!response.ok) {
    throw new Error('Network response was not ok');
  }
  return response.json();
};

function Map() {
  const [userLocation, setUserLocation] = useState(null);
  const [selectedAid, setSelectedAid] = useState('all');
  const [searchText, setSearchText] = useState('');
  const [page, setPage] = useState(1);
  const [selectedMarkerCoords, setSelectedMarkerCoords] = useState(null);
  const [categories, setCategories] = useState([
    'All Aids',
    'Basic Necessities Aid',
    'Healthcare Aid',
    'Education Aid',
    'Employment Aid',
    'Legal Aid',
    'Community Aid',
  ]);


  useEffect(() => {
    const getLocation = () => {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
          (position) => {
            setUserLocation([position.coords.latitude, position.coords.longitude]);
          },
          (error) => {
            console.error('Error getting user location:', error);
          },
          { enableHighAccuracy: true }
        );
      } else {
        console.error('Geolocation is not supported by this browser.');
      }
    };

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

```
    const { data, isLoading, isError } = useQuery('markers', fetchMarkers);

    if (isLoading) return <p>Loading...</p>;
    if (isError) return <p>Error fetching markers</p>;

    const filteredMarkers = data.filter(marker => {
      return (selectedAid === 'all' || marker.category_id === selectedAid) && marker.description.includes(searchText);
    });


    const calculateDistance = (markerCoords) => {
      if (userLocation) {
        const userLatLng = L.latLng(userLocation[0], userLocation[1]);
        const markerLatLng = L.latLng(markerCoords[0], markerCoords[1]);
        return userLatLng.distanceTo(markerLatLng) / 1000; // в метрах, переведемо у кілометри
      }
      return null;
    };

    const markersWithDistance = filteredMarkers.map((marker) => ({
      ...marker,
      distance: calculateDistance([marker.latitude, marker.longitude]),
    }));

    markersWithDistance.sort((a, b) => {
      if (a.distance === null) return 1;
      if (b.distance === null) return -1;
      return a.distance - b.distance;
    });

    // Відображення перших 10 маркерів
    const markersToShow = markersWithDistance.slice(0, page * 10);

    // Рендеринг компонента
    return (
      <>
        <div className="map">
          <div className="map__search">
            <div className="map__title"><strong style={{fontSize: 20}}>RefHelper</strong></div>
            <select
              className="form-select"
              style={{
                margin: "8px 0",
                padding: "10px",
                borderRadius: "4px",
                border: "1px solid #ced4da",
                backgroundColor: "#fff",
                color: "#495057",
                appearance: "none",
                backgroundImage: "url('data:image/svg+xml;utf8,<svg fill=%22%23333%22
xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 24 24%22><path d=%22M7 10l5 5 5-5H7z%22
/></svg>')", // зображення стрілки
                backgroundRepeat: "no-repeat",
                backgroundPosition: "right 8px center",
                backgroundSize: "24px",
                paddingLeft: "10px",
              }}
              id="markerColor"
              aria-label="Default select example"
              onChange={(e) => setSelectedAid(e.target.value)}
            >
```

```
                <option value="all">All Aids</option>
                <option value="2">Healthcare Aid</option>
                <option value="3">Education Aid</option>
                <option value="4">Employment Aid</option>
                <option value="5">Legal Aid</option>
                <option value="6">Community Aid</option>
                <option value="1">Basic Necessities Aid</option>
              </select>
              <input
                style={{
                  margin: "8px 0",
                  padding: "10px",
                  borderRadius: "4px",
                  border: "1px solid #ced4da",
                  backgroundColor: "#fff",
                  color: "#495057",
                  width: "100%",
                  boxSizing: "border-box",
                }}
                type="text"
                placeholder="Search"
                onChange={(e) => setSearchText(e.target.value)}
              />
              <ul className='map__places-list'>
                <p style={{fontSize: 18, letterSpacing: 0, lineHeight: 1}}>Results:</p>
                {markersToShow.map((marker, id) => (
                  <li onFocus={(e) => e.target.focus()}
                    tabIndex={0}
                    className='map__places-list-item'
                    key={id}
                    onClick={() => setSelectedMarkerCoords([marker.latitude, marker.longitude])}
                  >
                    <strong>{categories[marker.category_id]}</strong>
                    <p>{marker.description}</p>
                    <p>Distance: {marker.distance ? `${marker.distance.toFixed(2)} km` : 'Unknown'}</p>
                  </li>
                ))}
                {markersToShow.length < markersWithDistance.length && (
                  <button className='map__loadmore' onClick={() => setPage(page + 1)}>Load more</button>
                )}
              </ul>
            </div>
            {userLocation ? (
              <MapContainer center={userLocation} zoom={13} style={{ height: '100vh', width: '100%' }}>
                <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
                <ShowCrimes  data={data} selectedMarkerCoords={selectedMarkerCoords} />
              </MapContainer>
            ) : (
              <p>Loading...</p>
            )}
          </div>
        </>
    );
}

export default Map;
```

//Логіка кластеризації маркерів

```
// ShowCrimes.js
```

```
import React, { memo, useState, useEffect, useRef } from "react";
import L from "leaflet";
import "./ShowCrimes.css";
import useSupercluster from "use-supercluster";
import { Marker, useMap, Popup } from "react-leaflet";

const icons = {};
const fetchIcon = (count, newSize) => {
  let color;
  let size;
  if (count < 10) {
    color = 'green';
    size =  newSize;
  } else if (count < 100) {
    color = 'yellow';
    size =  newSize;
  } else {
    color = 'orange';
    size =  newSize;
  }

  if (!icons[count]) {
    icons[count] = L.divIcon({
      className: `cluster-marker ${color}`,
      html: `<div style="width: ${size}px; height: ${size}px;  display: flex; align-items: center; justify-content:
center">
    ${count}
   </div>`,
    });
  }
  return icons[count];
};

const cuffs = new L.Icon({
  iconUrl: 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/images/marker-icon-2x.png',
  iconSize: [21, 31], // size of the icon
});


function ShowCrimes ({ data, selectedMarkerCoords}) {
  const maxZoom = 22;
  const [bounds, setBounds] = useState(null);
  const [zoom, setZoom] = useState(12);
  const map = useMap();
  const updateMap = useRef();

  updateMap.current = () => {
    const b = map.getBounds();
    setBounds([
      b.getSouthWest().lng,
      b.getSouthWest().lat,
      b.getNorthEast().lng,
      b.getNorthEast().lat,
    ]);
    setZoom(map.getZoom());
  };

  useEffect(() => {
    const moveEndHandler = () => {
      updateMap.current();
```

```
    };

    map.on("moveend", moveEndHandler);

    return () => {
      map.off("moveend", moveEndHandler);
    };
}, []);



const points = data.map((crime) => ({
    type: "Feature",
    properties: { cluster: false, crimeId: crime.id, category: crime.category },
    geometry: {
      type: "Point",
      coordinates: [
        parseFloat(crime.longitude),
        parseFloat(crime.latitude),
      ],
    },
}));

const { clusters, supercluster } = useSupercluster({
    points: points,
    bounds: bounds,
    zoom: zoom,
    options: { radius: 75, maxZoom: 17 },
});

useEffect(() => {
    if (selectedMarkerCoords) {
      const [latitude, longitude] = selectedMarkerCoords;
      map.setView([latitude, longitude], 15, {
        animate: true,
      });
    }
}, [selectedMarkerCoords]);

return (
    <>
      {clusters.map((cluster, index) => {
        const { cluster: isCluster, point_count: pointCount } = cluster.properties;
        const [longitude, latitude] = cluster.geometry.coordinates;

        if (isCluster) {
          return (
            <Marker
              key={`cluster-${cluster.id}`}
              position={[latitude, longitude]}
              icon={fetchIcon(
                pointCount,
                10 + (pointCount / points.length) * 50
              )}
              eventHandlers={{
                click: () => {
                  const expansionZoom = Math.min(
                    supercluster.getClusterExpansionZoom(cluster.id),
                    maxZoom
                  );
                  map.setView([latitude, longitude], expansionZoom, {
```

```
                            animate: true,
                        });
                    },
                }}
            />
        );
    } else {

        const crime = data.find(crime => crime.id === cluster.properties.crimeId);
        return (
            <Marker
                key={`crime-${crime.id}-${index}`}
                position={[latitude, longitude]}
                icon={cuffs}
            >
                <Popup>
                    <div>
                        <h3>{crime.address}</h3>
                        <p>{crime.description}</p>
                        <p>Created Date: {crime.created_date}</p>
                    </div>
                </Popup>
            </Marker>
        );
    }
})}
</>
);
}

export default memo(ShowCrimes);
```

//Профіль користувача

```
import React, { useState, useEffect, useContext } from 'react';
import axios from 'axios';
import { AuthContext } from "../AuthContext/AuthProvider";
import './profile.css';
import AddOfferButton from "./addOfferButton/AddOfferButton";

const Profile = () => {
    const { user } = useContext(AuthContext);
    const [profileData, setProfileData] = useState(null);
    const [loading, setLoading] = useState(true);
    const [profile, setProfile] = useState(true);

    useEffect(() => {
        const fetchProfileData = async () => {
            if (user) {
                try {
                    const response = await axios.get('http://localhost:3001/auth/user', {
                        headers: {
                            'Authorization': `Bearer ${user.token}`,
                        },
                    });
                    return await response.data;
                } catch (error) {
                    console.error('Error fetching profile data:', error);
                }
            }
```

```
        };

        fetchProfileData().then(response => {
            setProfileData(response);
            setLoading(false);
        });
    }, [user]);

    const updateProfileData = (newProfileData) => {
        setProfileData(newProfileData);
    };

    if (!user) {
        return <p>Please sign in to view your profile.</p>;
    }

    if (loading) {
        return <p>Loading...</p>;
    }

    if (!profileData) {
        return <p>Profile data not found.</p>;
    }

    return (
        <div className={'profile'}>
            <div className={'container'}>
                <div className="profile__wrapper">
                    <h2 className='profile__title'>Home menu</h2>
                    <ul className='profile__menu'>
                        <li style={profile ? { fontWeight: 600, textDecoration: "underline" } : null} onClick={() =>
setProfile(true)}>My Profile</li>
                    <div></div>
                        {profileData.roles[0].toLowerCase() === 'volunteer' && <li style={!profile ? { fontWeight: 600,
textDecoration: "underline" } : null} onClick={() => setProfile(false)}>My Offered Aids</li>}
                    </ul>
                    {
                        profile ? <div className='profile__info'>
                                <p><strong>Name:</strong> {profileData.firstName} </p>
                                <p><strong>Surname:</strong> {profileData.lastName} </p>
                                <p><strong>Email:</strong> {profileData.email}</p>
                                <p><strong>Role:</strong> {profileData.roles[0].toLowerCase()}</p>
                            </div> :
                            <div className='profile__offers'>
                                <AddOfferButton user={user} onProfileUpdate={updateProfileData} />
                            </div>
                    }
                </div>
            </div>
        </div>
    );
};

export default Profile;
```

//Історія створених пропозицій

```
// AddOfferButton.jsx
import React, {useCallback, useEffect, useState} from 'react';
import Modal from 'react-modal';
```

```
import MapWithModal from '../mapWithModal/MapWithModal';
import axios from "axios";
import './addOfferButton.css'
import Slider from "react-slick"

const AddOfferButton = ({user}) => {
    const [modalIsOpen, setModalIsOpen] = useState(false);
    const [markers, setMarkers] = useState([]);
    const [userLocation, setUserLocation] = useState(null);
    const [markerLocation, setMarkerLocation] = useState(null)
    const [categories, setCategories] = useState([
        'All Aids',
        'Basic Necessities Aid',
        'Healthcare Aid',
        'Education Aid',
        'Employment Aid',
        'Legal Aid',
        'Community Aid',
    ]);

    const getMarker = useCallback(async () => {
        try {
            const response = await axios.get('http://localhost:3001/auth/user', {
                headers: {
                    'Authorization': `Bearer ${user.token}`,
                },
            });

            setMarkers(response.data.offersHistory)
        } catch (error) {
            console.error('Error fetching profile data:', error);
        }

    }, [user.token])

    useEffect(() => {
        Modal.setAppElement('#root');
        getMarker();
        navigator.geolocation.getCurrentPosition((position) => {
            setUserLocation([position.coords.latitude, position.coords.longitude]);
        });
    }, []);

    console.log(markers)

    const settings = {
        className: "center",
        centerMode: true,
        infinite: true,
        centerPadding: "0",
        slidesToShow: 1,
        speed: 500,
        rows: 2,
        slidesPerRow: 1,
    }

    const onDeleteOffer = async (e, _id) => {
        try {
            e.target.disabled = true;
            await axios.post(`http://localhost:3001/auth/deleteMarker`, { markerId: _id }, {
                headers: {
```

```
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${user.token}`,
            },
        });

        getMarker().then(() => e.target.disabled = false)
    }
    catch (error) {
        console.error('Error fetching profile data:', error);
    }
};

return (
    <div className='offers'>
        <ul className='offers__list'>
            <Slider {...settings}>
                {markers.map(item =>{
                    return (
                        <li

                            className='offers__list-item'
                            key={`list-${item._id}`}
                        >
                            <div className='offers__list-item-content'>
                                <p onClick={() => {
                                    setMarkerLocation([item.latitude, item.longitude])
                                    setModalIsOpen(true)
                                }} className='offers__list-item-title'>{categories[item.category_id]}:</p>
                                <p><span>Address:</span> {item.address}</p>
                                <p><span>Date of creation:</span>{item.type}</p>
                            </div>
                            <button onClick={(e) => onDeleteOffer(e, item._id)} className='offer__list-delete-
btn'>&#215;</button>
                        </li>
                    )
                })}
            </Slider>

        </ul>

        <button
            className={'offers__add-btn'}
            onClick={() => {
            setModalIsOpen(true);
            setMarkerLocation(null);
        }}>Add Offer</button>
        <Modal  isOpen={modalIsOpen} onRequestClose={() => setModalIsOpen(false)}>
            <MapWithModal markerLocation={markerLocation} location={userLocation} user={user}
markers={markers} getMarkers={getMarker}/>
        </Modal>
    </div>
);
};

export default AddOfferButton;
```

//Карта волонтера

```
import React, {useState, useRef, useCallback} from 'react';
```

```
import { MapContainer, TileLayer, useMapEvents, Marker, Popup } from 'react-leaflet';
import Modal from 'react-modal';
import loop from './loop.png'
import marker from './marker.png'
import axios from "axios";
import L from "leaflet";

Modal.setAppElement('#root');

const MapWithModal = React.memo(({markers, getMarkers, user, location, markerLocation}) => {
   const [modalIsOpen, setModalIsOpen] = useState(false);
   const [zoomMode, setZoomMode] = useState(true);
   const [buttonClicked, setButtonClicked] = useState(false);
   const [formData, setFormData] = useState({
      type: '',
      address: '',
      description: '',
      latitude: '',
      longitude: '',
      category_id: '2',
   });

   const [categories, setCategories] = useState([
      'All Aids',
      'Basic Necessities Aid',
      'Healthcare Aid',
      'Education Aid',
      'Employment Aid',
      'Legal Aid',
      'Community Aid',
   ]);


   const lastClickRef = useRef();

   const MapEvents = () => {
      const map = useMapEvents({
         click: (e) => {
            if (buttonClicked) {
               setButtonClicked(false);
               return;
            }

            const { lat, lng } = e.latlng;
            const type = new Date().toLocaleString();
            setFormData({ ...formData, latitude: lat, longitude: lng, type});

            if (!zoomMode) {
               const now = Date.now();
               const DOUBLE_CLICK_THRESHOLD = 100;

               if (lastClickRef.current && now - lastClickRef.current < DOUBLE_CLICK_THRESHOLD) {

                  return;
               }

               lastClickRef.current = now;
               setModalIsOpen(true);
            }
         },
      });
```

```
    return null;
  };

  const closeModal = useCallback(() => {
    setModalIsOpen(false);
  }, []);

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

  const handleSubmit = useCallback(async (e) => {
    e.preventDefault();

    try {
      const response = await axios.post('http://localhost:3001/auth/markers', formData, {
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${user.token}`,
        }
      });

      if (response.status === 201) {
        console.log('Offer history added successfully');
      }
      else {
        console.error('Error adding offer history:', response.statusText);
      }

    } catch (error) {
      console.error('Error adding offer history:', error.message);
    }


    getMarkers();
    closeModal();
    setFormData({
      type: '',
      address: '',
      description: '',
      latitude: '',
      longitude: '',
      category_id: '2',
    })
  },[user.token, formData, getMarkers, closeModal])

  const cuffs = new L.Icon({
    iconUrl: 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/images/marker-icon-2x.png',
    iconSize: [17, 26], // size of the icon
  });

  return (
    <>
      <div style={{width: '100%', height: '100%', position: 'relative'}}>
        <MapContainer center={!markerLocation ? location : markerLocation} zoom={markerLocation ? 50 : 13}
style={{ height: '100%', width: '100%' }} doubleClickZoom={zoomMode}>
          <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
          <MapEvents />
          {markers?.map((item) =>
            <Marker icon={cuffs} key={`marker-${item._id}`} position={[item.latitude, item.longitude]}>
```

```
                    <Popup>
                      <div>
                        <h3>{categories[item.category_id]}</h3>
                        <p>{item.address}</p>
                        <p>{item.description}</p>
                        <p>Created Date: {item.type}</p>
                      </div>
                    </Popup>
                  </Marker>
                )}
              </MapContainer>
              <button style={{cursor:"pointer", position: 'absolute', top: '15px', right: 12, zIndex: '1000', width: 30, height: 30, border: '1px solid white', background: 'white'}} onClick={(e) => { e.preventDefault(); setZoomMode(!zoomMode); setButtonClicked(true); }}>
                {!zoomMode ? <img style={{width: 25, height: 25, paddingTop: 2}} src={marker} alt=""/> :
                  <img style={{width: 25, height: 25, paddingTop: 2}} src={loop} alt=""/>}
              </button>
            </div>

            <Modal className='offers__modal' isOpen={modalIsOpen} onRequestClose={closeModal}>
              <h2 className='offers__modal-title'>Add an offer</h2>
              <form className='offers__form' onSubmit={handleSubmit}>
                <label>
                  Address:
                  <input type="text" required name="address" value={formData.address} onChange={handleChange} />
                </label>
                <label>
                  Description:
                  <textarea required name="description" value={formData.description} onChange={handleChange} cols="30" rows="10"></textarea>
                </label>
                <label>
                  Category:
                  <select name="category_id" value={formData.category_id} onChange={handleChange}>
                    <option value="2">Healthcare Aid</option>
                    <option value="3">Education Aid</option>
                    <option value="4">Employment Aid</option>
                    <option value="5">Legal Aid</option>
                    <option value="6">Community Aid</option>
                    <option value="1">Basic Necessities Aid</option>
                  </select>
                </label>
                <button className='offers__modal-create-btn' type="submit">Create</button>
              </form>
            </Modal>
          </>
        );
});

export default MapWithModal;
```

//AuthController

```
const User = require('./models/User')
const Role = require('./models/Role')
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { validationResult } = require('express-validator')
const { secret } = require("./config")
const OfferHistory = require("./models/offerHistorySchema");
```

```
const {addMarkerToFakeOffers, deleteMarkerFromFile} = require("./routes/api");

const generateAccessToken = (id, roles) => {
   const payload = {
      id,
      roles
   }
   return jwt.sign(payload, secret, { expiresIn: "24h" })
}

class authController {
   async registration(req, res) {
      try {
         const errors = validationResult(req)
         if (!errors.isEmpty()) {
            return res.status(400).json({ message: "При вході сталася помилка. Перевірте введену адресу
електронної пошти та пароль або створіть обліковий запис.", errors })
         }
         const { firstName, lastName, email, password } = req.body;
         const candidate = await User.findOne({ email })
         if (candidate) {
            return res.status(400).json({ message: "Користувач з такою адресою існує" })
         }
         const hashPassword = bcrypt.hashSync(password, 7);
         const userRole = await Role.findOne({value: "REFUGEE"})
         const user = new User({ firstName, lastName, email, password: hashPassword, roles: [userRole.value]})
         await user.save()
         return res.json({ message: "Користувач успішно зареєстрований" })
      } catch (e) {
         console.log(e)
         res.status(400).json({ message: 'Registration error' })
      }
   }

   async login(req, res) {
      try {
         const { firstName, lastName, email, password } = req.body
         const user = await User.findOne({ email })
         if (!user) {
            return res.status(400).json({ message: `Користувача ${email} не знайдено` })
         }
         const validPassword = bcrypt.compareSync(password, user.password)
         if (!validPassword) {
            return res.status(400).json({ message: `Введен неверный пароль` })
         }
         const token = generateAccessToken(user._id, user.roles)
         return res.json({ token })
      } catch (e) {
         console.log(e)
         res.status(400).json({ message: 'Login error' })
      }
   }

   async getUser(req, res) {
      try {
         const token = req.headers.authorization.split(' ')[1]
         if (!token) {
            return res.status(403).json({ message: "У вас немає доступа" })
         }
         const { id } = jwt.verify(token, secret)
         const user = await User.findById(id)
```

```
        if (!user) {
            return res.status(404).json({ message: "Користувача з таким id не знайдено" })
        }
        res.json(user);
    } catch (e) {
        console.log(e)
        res.status(500).json({ message: 'Internal Server Error' })
    }
}

async addOfferHistory(req, res) {
    try {
        const token = req.headers.authorization.split(' ')[1];
        const { type, address, description, latitude, longitude, category_id } = req.body;
        const { id } = jwt.verify(token, secret);

        // Пошук користувача в базі даних
        const user = await User.findById(id);
        if (!user) {
            return res.status(404).json({ message: 'User not found' });
        }

        // Створення нового об'єкта пропозиції
        const newOfferHistory = new OfferHistory({
            type,
            address,
            description,
            latitude,
            longitude,
            category_id
        });

        user.offersHistory.push(newOfferHistory);

        await user.save();

        const latestOffer = user.offersHistory[user.offersHistory.length - 1];

        addMarkerToFakeOffers(latestOffer);

        res.status(201).json({ message: 'Offer history added successfully', offerHistory: newOfferHistory });
    } catch (error) {
        console.error('Error adding offer history:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
}

async deleteOfferFromHistory(req, res) {
    try {
        const token = req.headers.authorization.split(' ')[1];

        const { id } = jwt.verify(token, secret);

        const { markerId } = req.body;

        if (!markerId){
            return res.status(404).json({ message: 'MarkerId not found' } );
        }

        const user = await User.findById(id);
```

```
            if (!user) {
                return res.status(404).json({ message: 'User not found' });
            }

            const index = user.offersHistory.findIndex((offer) => offer._id.toString() === markerId);
            if (index !== -1) {
                user.offersHistory.splice(index, 1);
            } else {
                return res.status(404).json({ message: 'Offer not found in user\'s history' });
            }

            await user.save();
            deleteMarkerFromFile(markerId)

            res.status(200).json({ message: 'Offer deleted successfully' });
        } catch (error) {
            console.error('Error deleting offer from history:', error);
            res.status(500).json({ message: 'Internal Server Error' });
        }
    }


}

module.exports = new authController()
```

//AuthRouter

```
const express = require('express');
const { check } = require("express-validator");
const authController = require('./authController');
const roleMiddleware = require('./middlewaree/roleMiddleware')

const router = express.Router();

router.post('/registration', [
    check('firstName', "Ім'я користувача не може бути порожнім").notEmpty(),
    check('lastName', "Прізвище користувача не може бути порожнім").notEmpty(),
    check('email', "Введіть коректну адресу електронної пошти").isEmail(),
    check('password', "Пароль має містити від 6 до 20 символів").isLength({ min: 6, max: 20 })
], authController.registration);

router.post('/login', authController.login);
router.get('/user', roleMiddleware(['VOLUNTEER', 'REFUGEE']), authController.getUser);
router.post('/markers', authController.addOfferHistory);
router.post('/deleteMarker', authController.deleteOfferFromHistory);
module.exports = router;

const jwt = require("jsonwebtoken");
const {secret} = require("../config");
module.exports =  function (roles){
    return function (req, res, next){
        if (req.method === "OPTIONS") {
            next()
        }

        try {
            const token = req.headers.authorization.split(' ')[1]
            if (!token) {
                return res.status(403).json({ message: "У вас немає доступа" })
```

```
            }
            const {roles: userRoles} = jwt.verify(token, secret)
            let hasRole = false
            userRoles.forEach(role => {
                if (roles.includes(role)){
                    hasRole = true
                }
            })
            if(!hasRole){
                return res.status(403).json({ message: "У вас немає доступа" })

            }
            next()
        } catch (e) {
            console.log(e)
            return res.status(403).json({ message: "Користувач не авторизований" })
        }
    }
}
```

Додаток Г – Ілюстративна частина

РОЗРОБКА ІНТЕГРОВАНОГО ОНЛАЙН-ПОРТАЛУ ДЛЯ БІЖЕНЦІВ З ПЕРСОНАЛІЗОВАНИМИ РЕКОМЕНДАЦІЯМИ

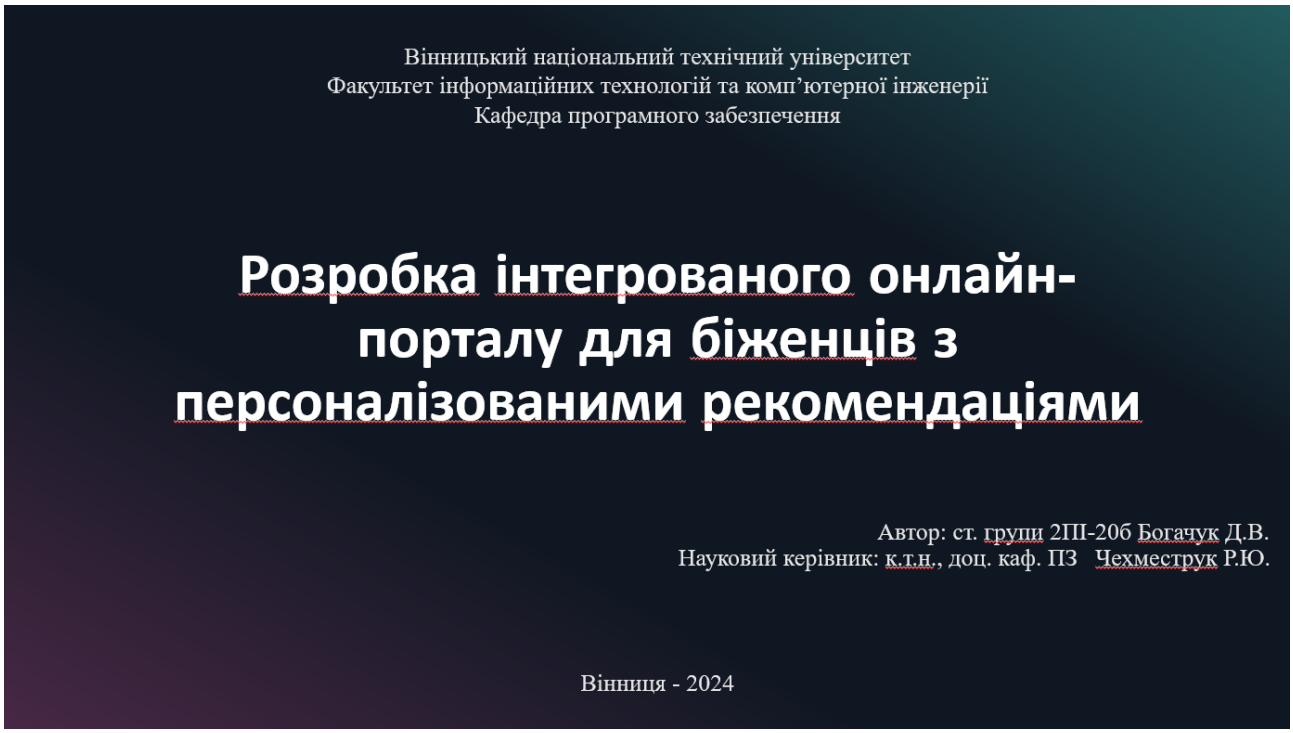

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

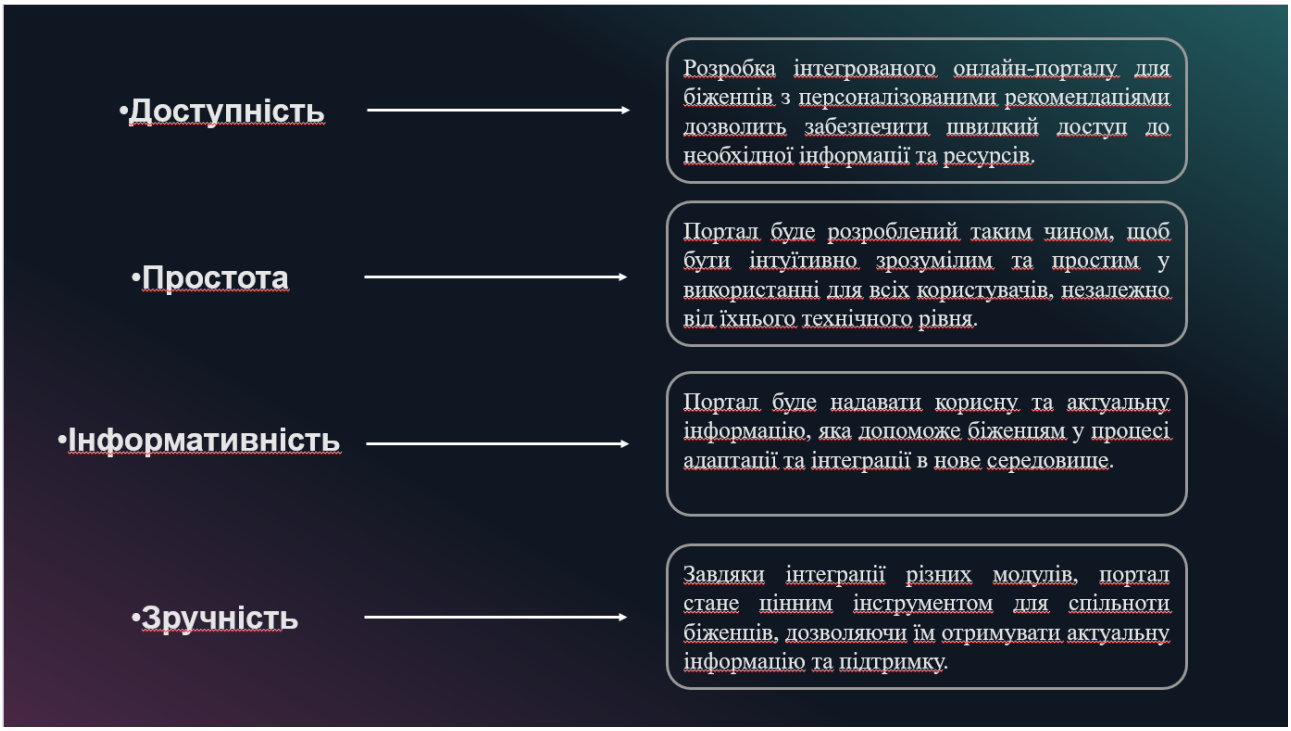

Рисунок Г.3 – Доступність, простота, інформативність та зручність веб-застосунку

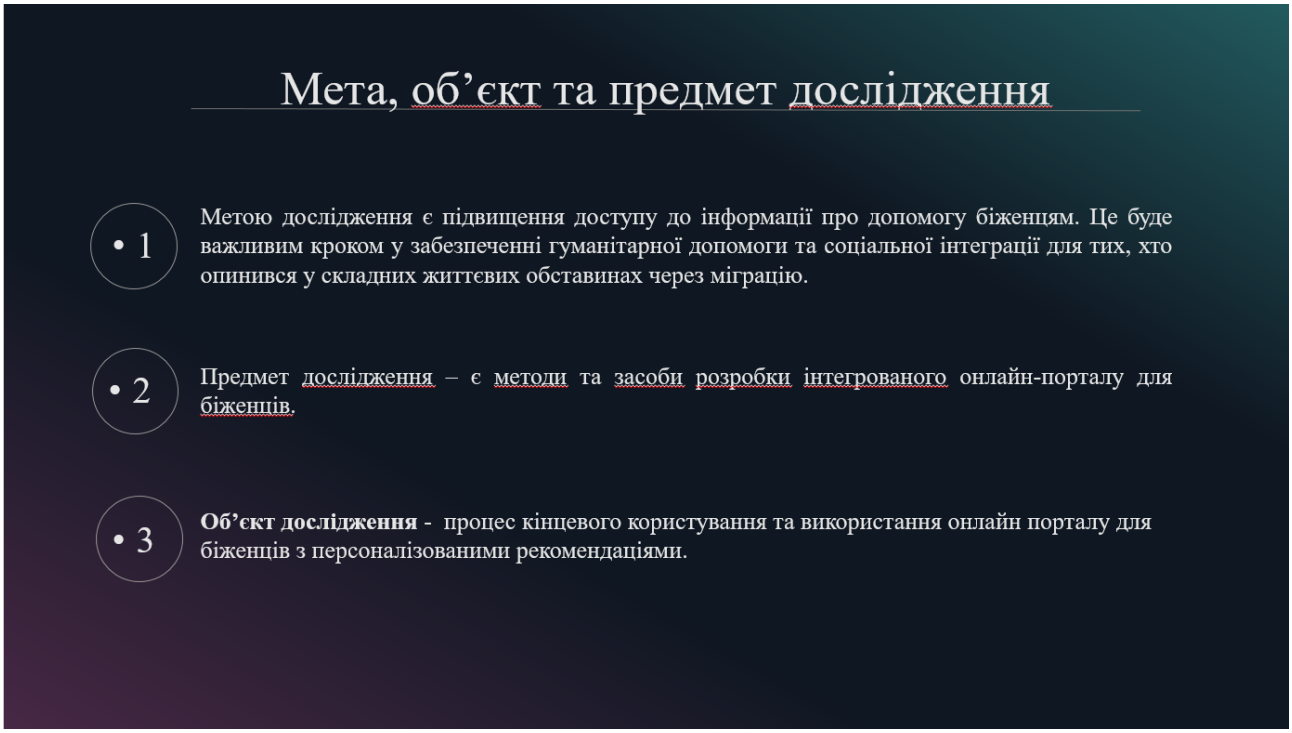

Рисунок Г.4 – Мета, об’єкт та предмет дослідження

Рисунок Г.5 – Завдання дослідження

Рисунок Г.6 – Наукова новизна

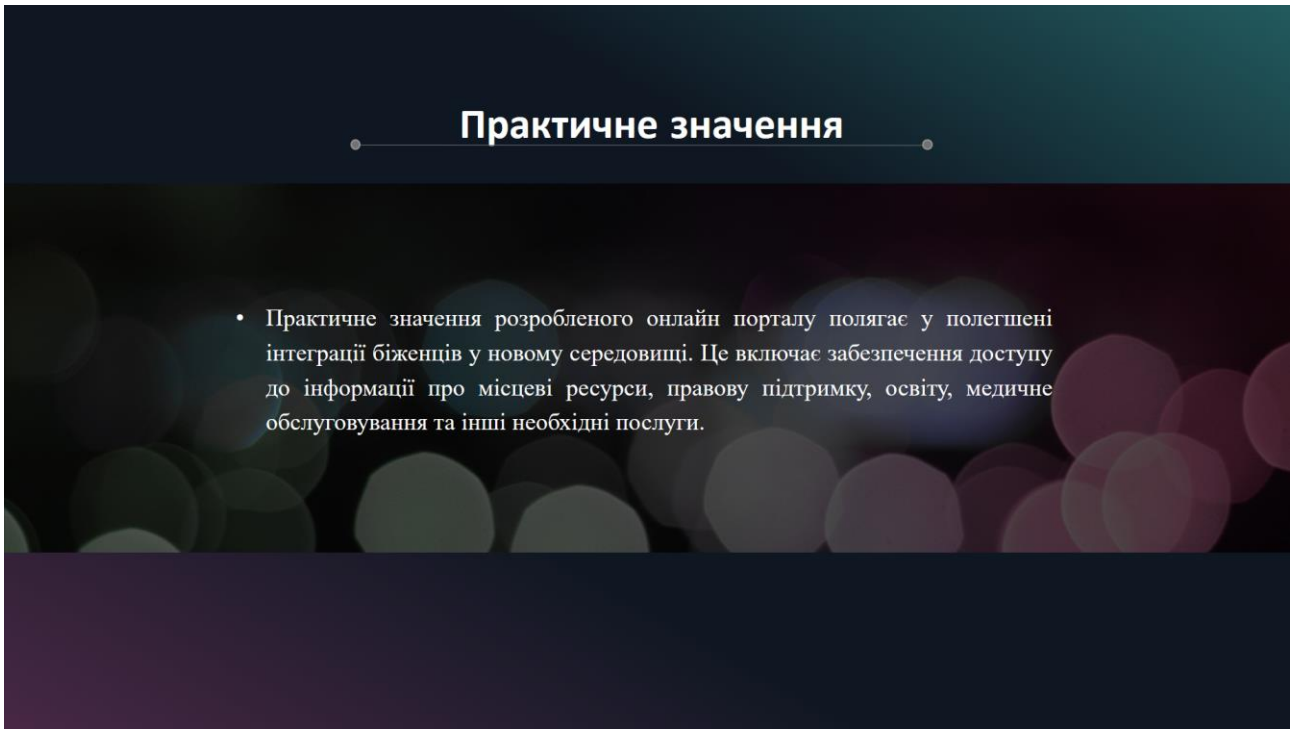

Рисунок Г.7 – Практична цінність

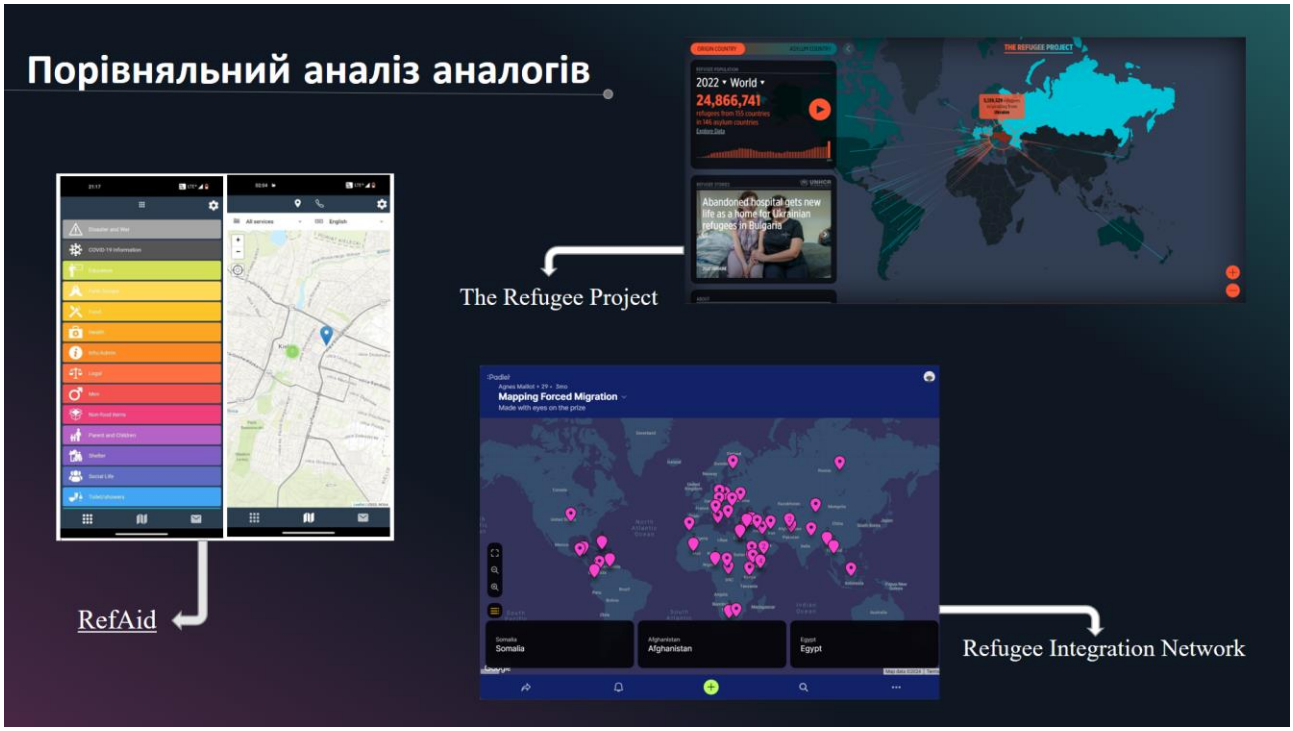

Рисунок Г.8 – Аналіз аналогів (частина 1)

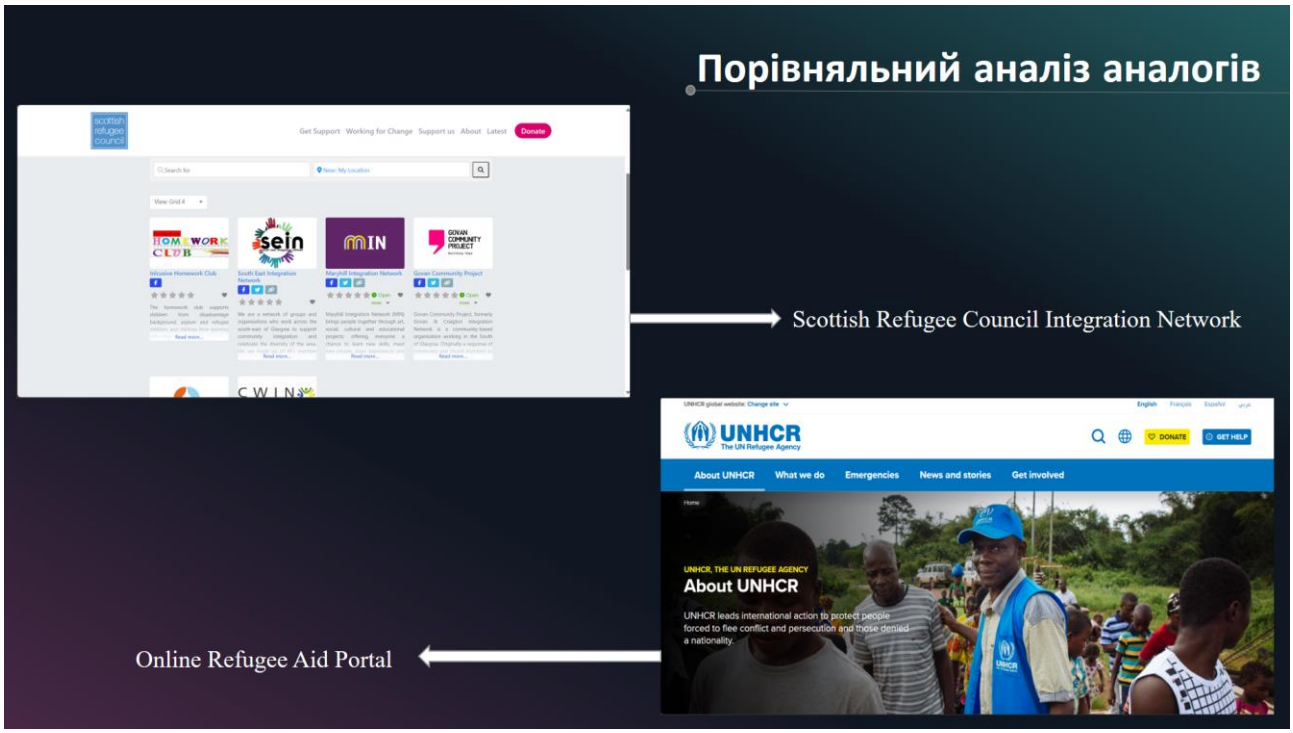

Рисунок Г.9 – Аналіз аналогів (частина 2)

Порівняльний аналіз аналогів

Критерії	Online Refugee Aid Portal	The Refugee Project	Refugee Integration Network	RefAid	Refugee Council Integration Network	Власний модуль
Доступність інформації та ресурсів	+	+	+	+	+	+
Можливість інтерактивної взаємодії	-	+	+	+	-	+
Підтримка адаптації та отримання підтримки	+	+	+	+	+	+
Швидкість та зручність використання	+	+	+	+	+	+
Модуль з новинами для додаткового отримання інформації	+	-	-	-	-	+
Загальна оцінка	80%	80%	80%	80%	60%	100%

Рисунок Г.10 – Порівняльний аналіз аналогів (частина 3)

Рисунок Г.11 – Технології розробки

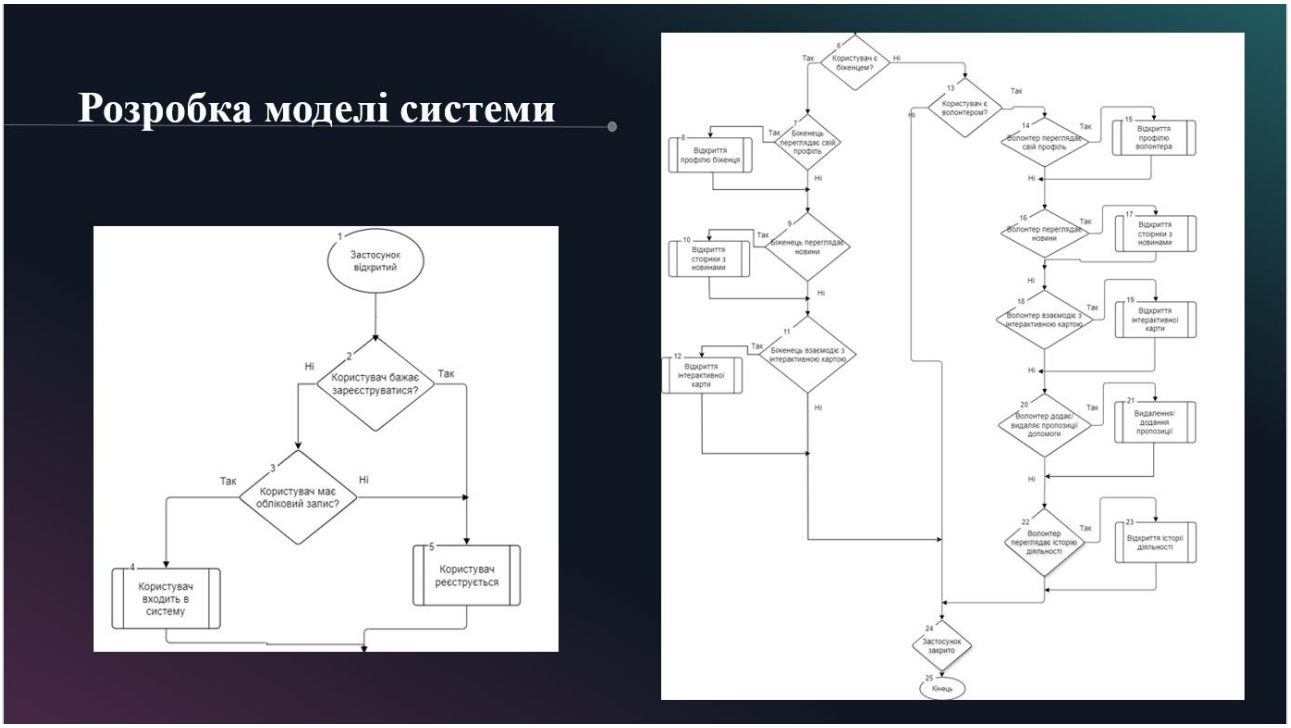

Рисунок Г.12 – Загальна модель

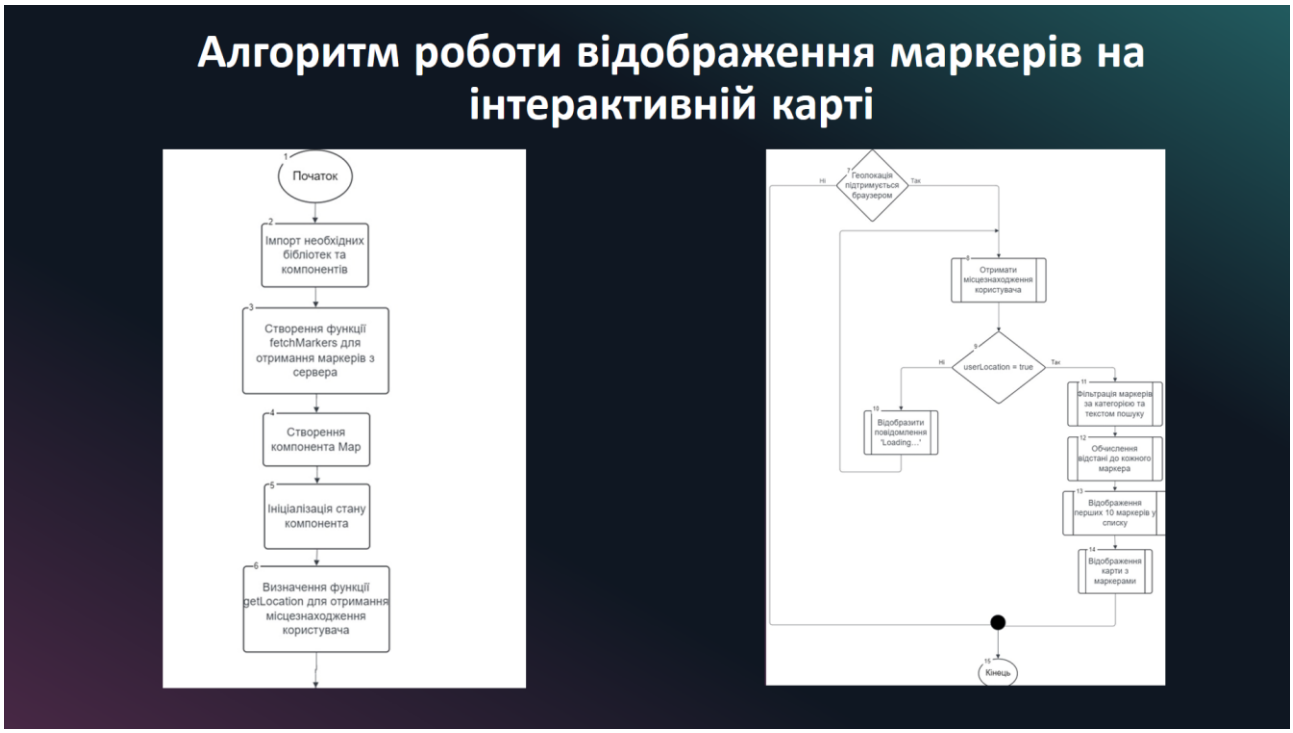

Рисунок Г.13 – Алгоритм роботи відображення маркерів на інтерактивній карті (частина 1)

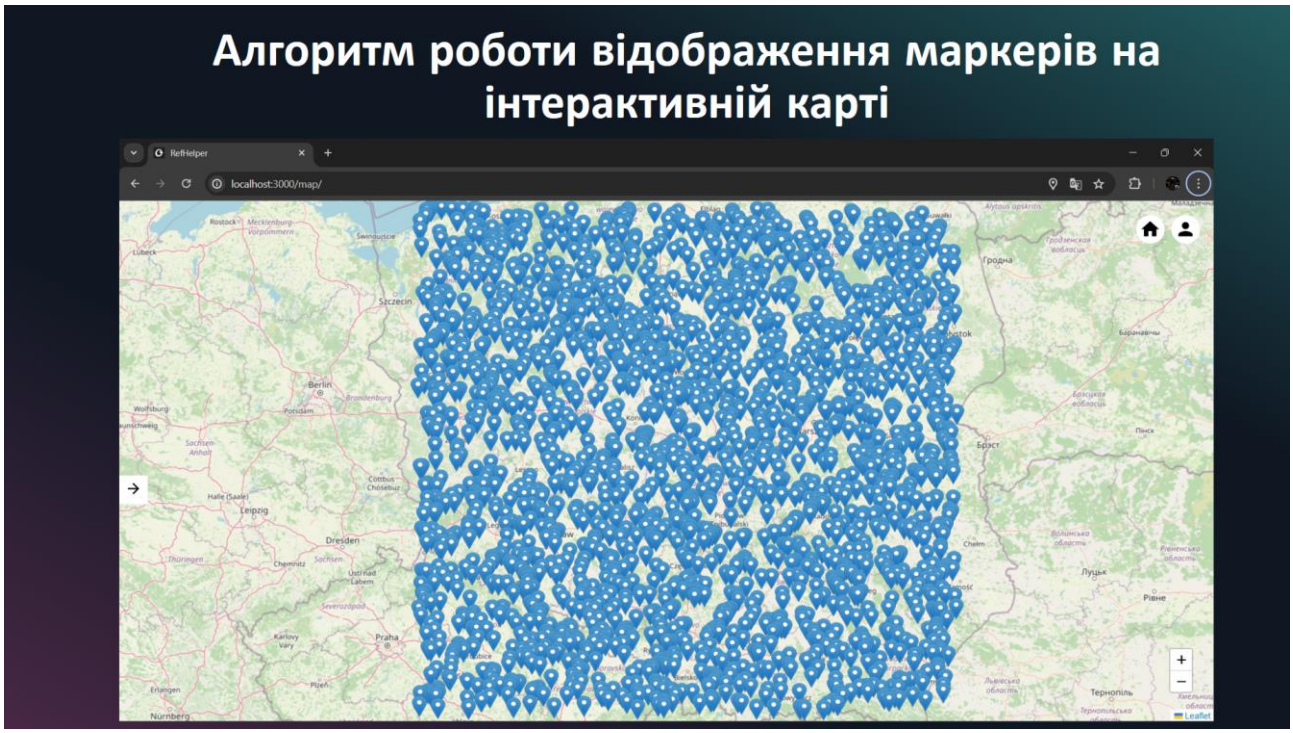

Рисунок Г.14 – Алгоритм роботи відображення маркерів на інтерактивній карті (частина 2)

Рисунок Г.15 – Алгоритм роботи кластеризації маркерів (частина 1)

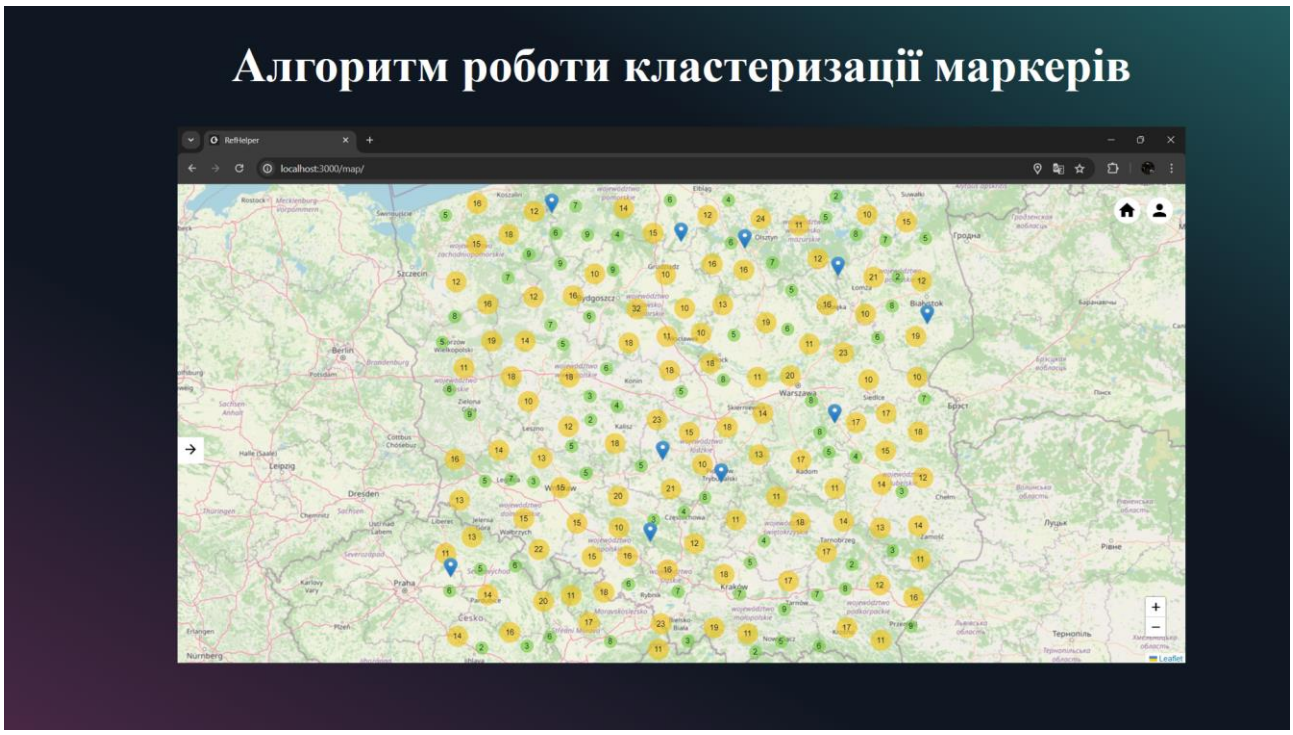

Рисунок Г.16 – Алгоритм роботи кластеризації маркерів (частина 2)

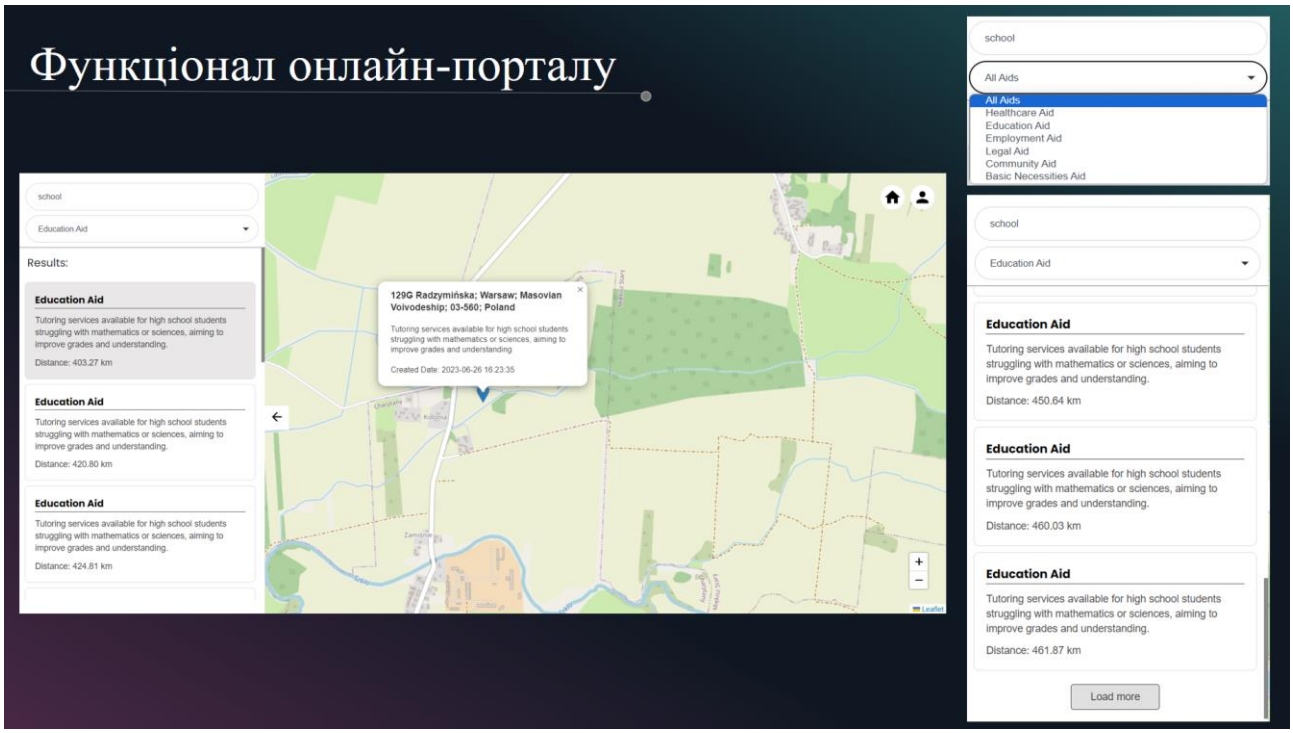

Рисунок Г.17 – Функціонал онлайн-порталу

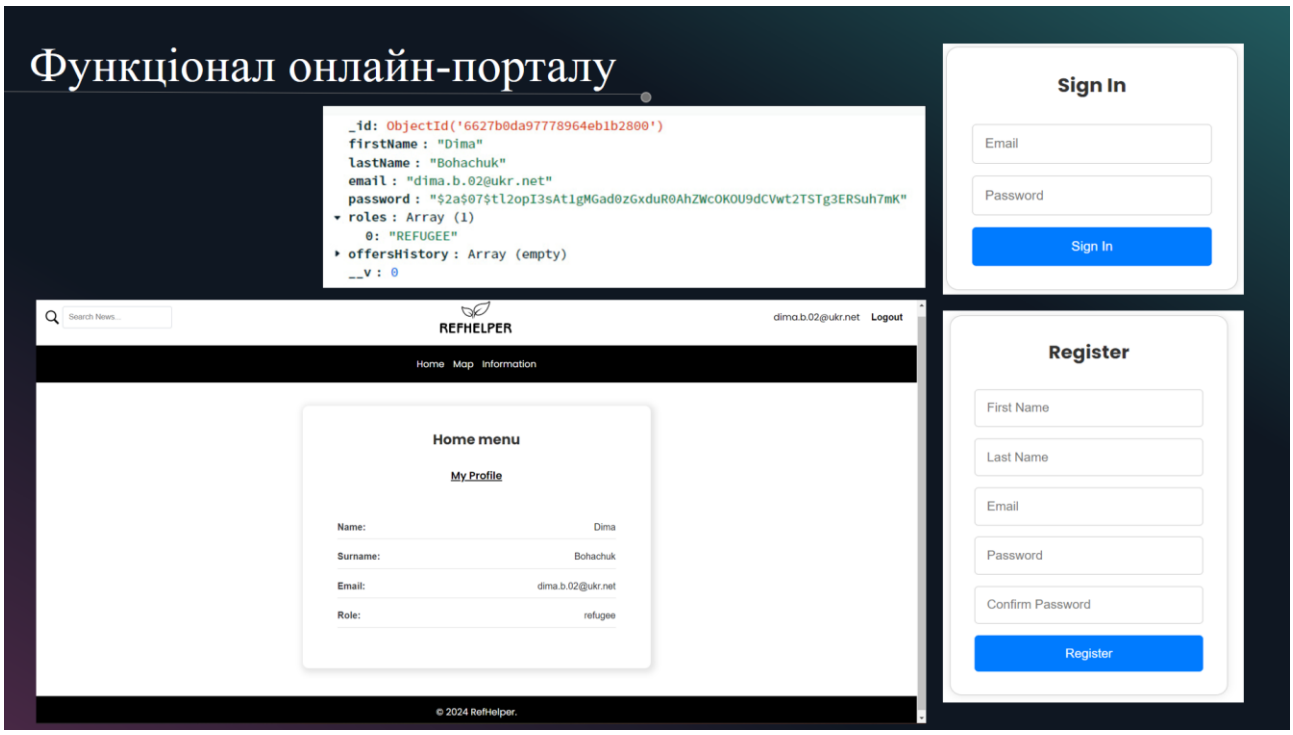

Рисунок Г.18 – Функціонал онлайн-порталу

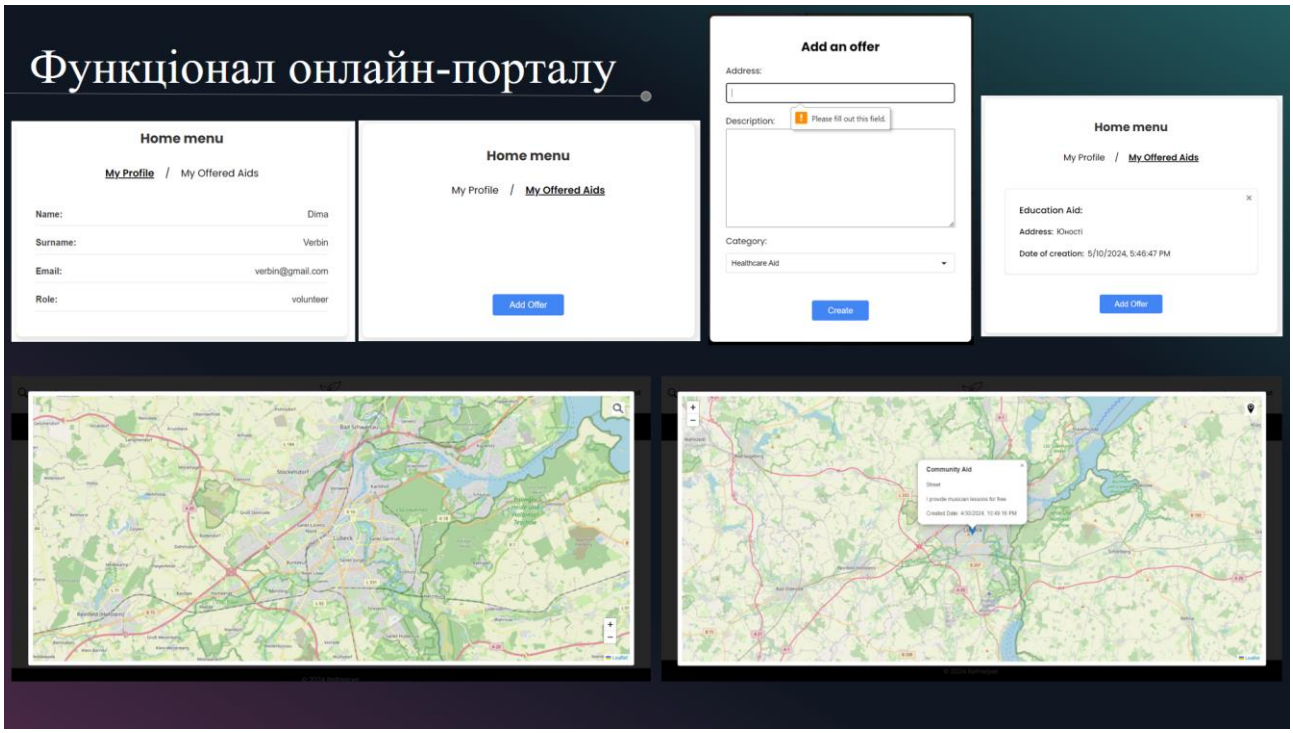

Рисунок Г.19 – Функціонал онлайн-порталу

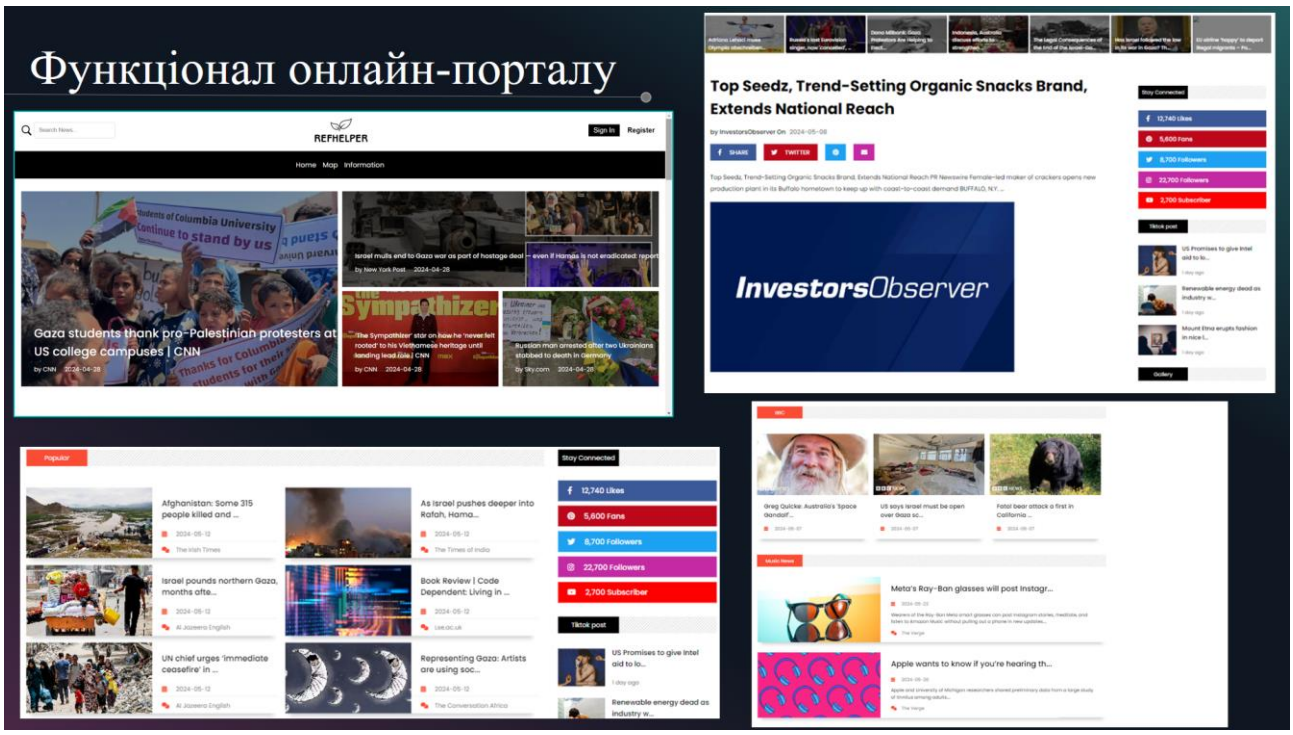

Рисунок Г.20 – Функціонал онлайн-порталу

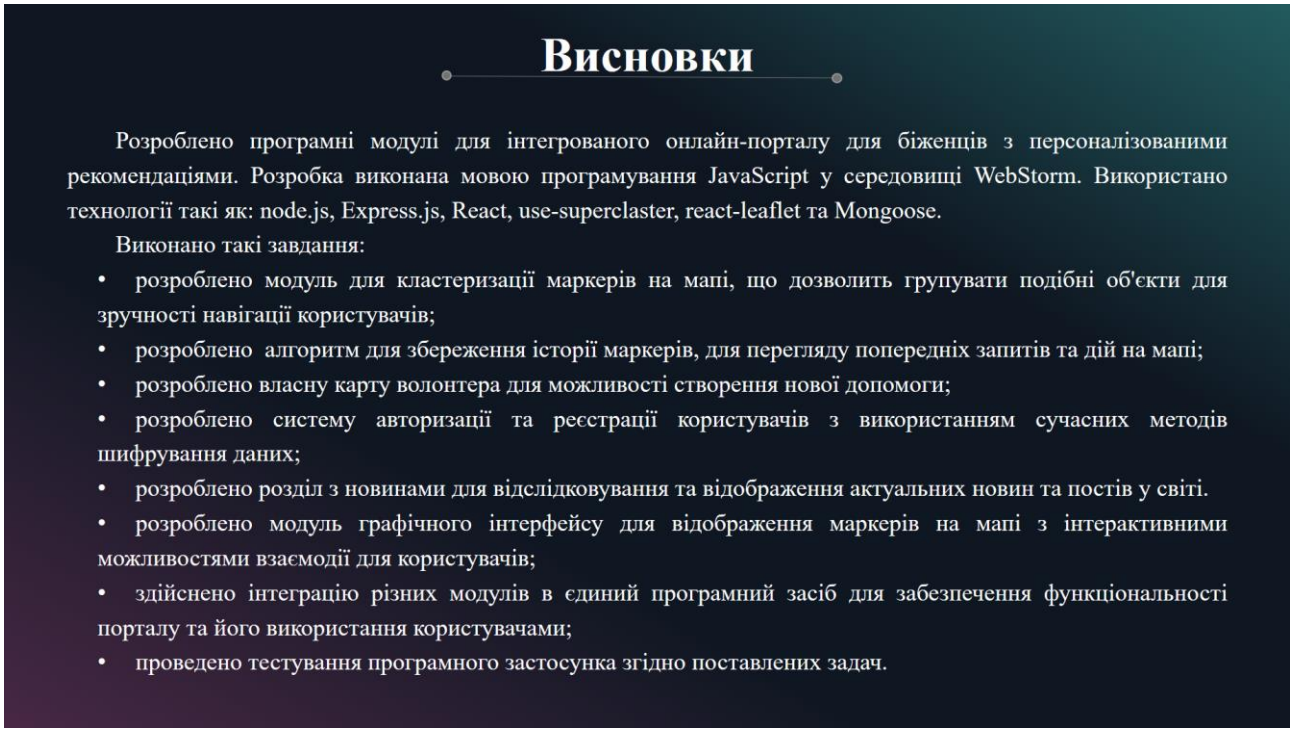

Рисунок Г.21 – Висновки

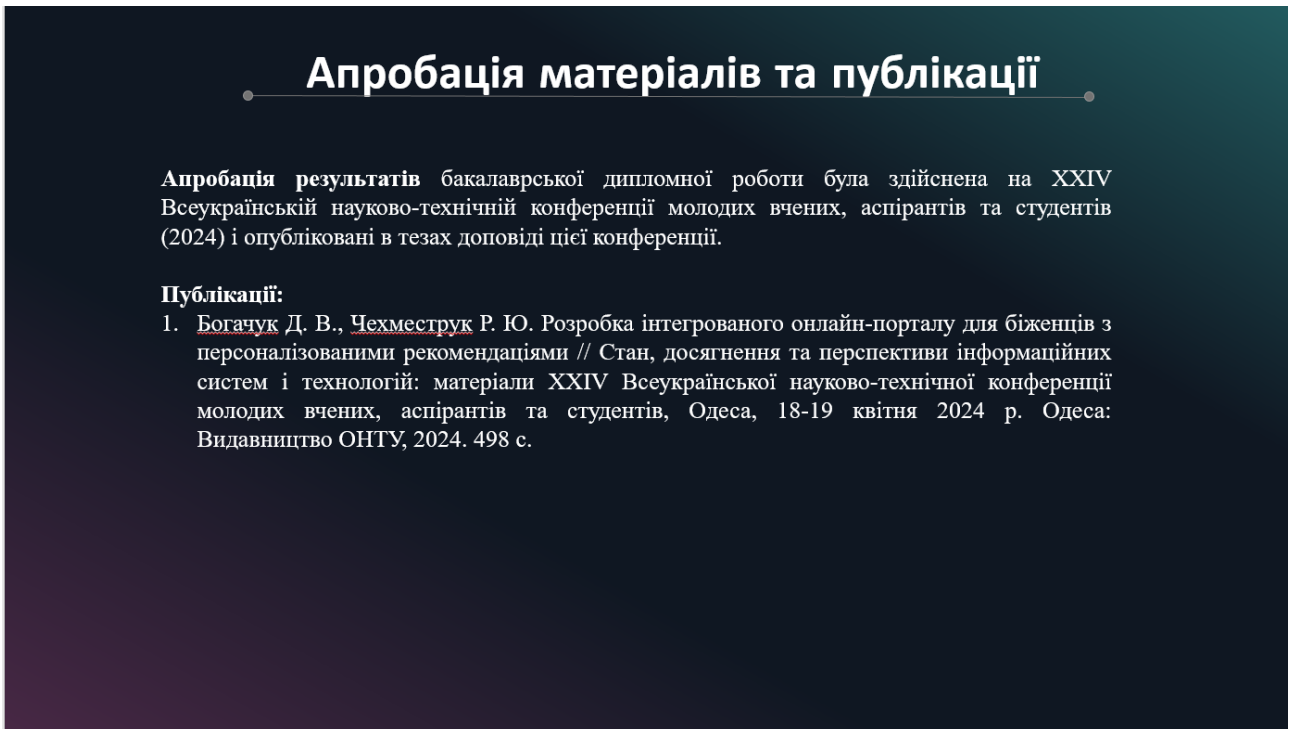

Рисунок Г.22 – Апробація матеріалів та публікації