**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра програмування

(повна назва кафедри)

**КУРСОВА РОБОТА**

"Розробка мобільного застосунку для допомоги людям з вадами зору"

Виконав: студент групи ПМІ-36 спеціальності 122 – комп'ютерні науки

(шифр і назва спеціальності)

Назар Проць

(підпис) (прізвище та ініціали)

Керівник Сергій Ярошко .

(підпис) (прізвище та ініціали)

2025

**Зміст**

[ВСТУП 3](#_Toc199084879)

[1 АНАЛІЗ КОНКУРЕНТІВ 4](#_Toc199084880)

[1.1 Загальні переваги 4](#_Toc199084881)

[1.2 Загальні недоліки 4](#_Toc199084882)

[1.3 Цільова аудиторія 5](#_Toc199084883)

[2 ВИМОГИ ДО ПРОЄКТУ 6](#_Toc199084884)

[2.1 Функціональні вимоги 6](#_Toc199084885)

[2.2 Нефункціональні вимоги 6](#_Toc199084886)

[3 ОПИС СИСТЕМИ 8](#_Toc199084887)

[3.1 Архітектура проєкту 8](#_Toc199084888)

[3.2 Використані технології 9](#_Toc199084889)

[4 ПРОГРАМНА РЕАЛІЗАЦІЯ 12](#_Toc199084890)

[4.1 Серверна частина 12](#_Toc199084891)

[4.1.1 Модуль авторизації та автентифікації 12](#_Toc199084892)

[4.1.2 Модуль опрацювання локації користувача 12](#_Toc199084893)

[4.1.3 Модуль виклику екстрених служб 13](#_Toc199084894)

[4.1.4 Модуль надсилання сповіщення довіреним особам 14](#_Toc199084895)

[4.1.5 Модуль для забезпечення багатомовності 15](#_Toc199084896)

[4.1.6 Модуль для обробки винятків 16](#_Toc199084897)

[4.1.7 Контролер для мультимодального розпізнавання 16](#_Toc199084898)

[4.2 Модуль мультимодального розпізнавання 16](#_Toc199084899)

[4.3 Клієнтська частина 18](#_Toc199084900)

[4.3.1 Сервіс для надсилання запитів до серверу 18](#_Toc199084901)

[4.3.2 Модуль голосової взаємодії з користувачем: 18](#_Toc199084902)

[4.3.3 Модуль тактильних відкликів 18](#_Toc199084903)

[4.3.4 Модуль для опрацювання локації користувача 19](#_Toc199084904)

[4.3.5 Модуль для аналізу об’єктів 19](#_Toc199084905)

[4.3.6 Модуль для забезпечення багатомовності 20](#_Toc199084906)

[4.3.6 Навігація між екранами 20](#_Toc199084907)

[5 ТЕСТУВАННЯ СИСТЕМИ 21](#_Toc199084908)

[6 ОГЛЯД РЕЗУЛЬТАТІВ 22](#_Toc199084909)

[ВИСНОВОК 26](#_Toc199084910)

[СПИСОК ДЖЕРЕЛ 27](#_Toc199084911)

ВСТУП

У сучасному світі технології відіграють важливу роль у житті кожної людини, забезпечуючи доступ до різноманітної інформації, спрощуючи комунікацію та відкриваючи нові можливості для взаємодії з навколишнім середовищем. Особливо актуальним є створення програмного забезпечення, яке сприяє покращенню якості життя людей з особливими потребами, зокрема, осіб з вадами зору. Вони стикаються з численними труднощами, де обмеження візуального сприйняття значно ускладнює виконання звичних завдань. Метою цієї роботи є закласти основу в розробці мобільного застосунку, який підвищить рівень автономності людей з вадами зору, а також допоможе як у критичних, так і в повсякденних ситуаціях. Для досягнення цієї мети були визначенні певні завдання, а саме:

* Розробка функціоналу для аналізу предметів навколишнього середовища.
* Розробка функціоналу для швидкого виклику екстрених служб або надсилання сповіщення контактним особам у разі надзвичайної ситуації.
* Розробка інтуїтивно зрозумілого інтерфейсу з голосовим супроводом та тактильними відгуками.
* Інтеграція із системами навігацій.
* Створення бази даних важливих контактів і ресурсів для осіб з вадами зору.
* Забезпечення доступності застосунку для різних мов.

Ця робота також передбачала тісну співпрацю з кінцевими користувачами – осіб з вадами зору, що дозволило отримати зворотний зв’язок та адаптувати функціонал під їхні потреби. Таким чином, основна ціль цієї роботи – створити програмне рішення, яке допоможе людям з вадами зору мати незалежне, повноцінне та безпечне життя.

1 АНАЛІЗ КОНКУРЕНТІВ

Під час розробки мобільного застосунку було проаналізовано функціональні можливості вже наявних технічних рішень, а також проведено опитування представників цільової аудиторії для визначення найбільш популярних та ефективних інструментів. У результаті дослідження було виокремлено два застосунки: Be My Eyes та Envision.



Рисунок 1.1 – Застосунки Be My Eyes та Envision

1.1 Загальні переваги

Як було описано вище, ці застосунки є найбільш популярними та релевантними для покращення якості життя людей з вадами зору. Їх спільними характеристиками є зручний інтерфейс, доступність на найпопулярніших мобільних платформах, а також наявність функціоналу, що дозволяє користувачам отримати допомогу в режимі реального часу – як від волонтерів (у випадку Be My Eyes), так і за допомогою систем штучного інтелекту та розпізнавання об’єктів (у випадку Envision та Be My Eyes).

1.2 Загальні недоліки

Спільним недоліком обох застосунків є відсутність функціоналу для допомоги в екстрених ситуаціях. Окрім того, повний функціонал застосунку Envision доступний лише у платній версії, а Be My Eyes значною мірою залежить від участі зрячих волонтерів. Іншим спільним недоліком є відсутність інтегрованих картографічних сервісів для визначення поточного місцезнаходження користувача або пошуку необхідних об’єктів поблизу.

1.3 Цільова аудиторія

Застосунки Be My Eyes та Envision розраховані на широку аудиторію осіб з вадами зору, які мають базові навички користування мобільним пристроєм. Be My Eyes переважно використовується людьми, яким потрібна допомога в повсякденних ситуаціях і які надають перевагу взаємодії з реальними волонтерами. Натомість Envision більше підходить користувачам, які шукають повної автономності та готові покладатись на засоби штучного інтелекту для розпізнавання об’єктів, тексту чи аналізу навколишнього середовища загалом.

2 ВИМОГИ ДО ПРОЄКТУ

Для визначення очікувань користувачів щодо майбутнього застосунку, було проведено опитування людей з порушеннями зору та проаналізовано наявні програмні рішення. На основі зібраних даних сформовано перелік функціональних і нефункціональних вимог, реалізація яких забезпечить зручність користування, ефективність роботи застосунку, а також відповідатиме вимогам цільової аудиторії.

2.1 Функціональні вимоги

* + Надання голосових і тактильних підказок при взаємодії з інтерфейсом, зокрема при активації елементів управління та використанні функціоналу.
  + Надсилання сповіщення довіреним особам у разі надзвичайних ситуацій.
  + Інтеграція з картографічними сервісами для орієнтації користувача в просторі, зокрема для визначення поточного місцезнаходження та пошуку найближчих об’єктів інфраструктури.
  + Розпізнавання об’єктів навколишнього середовища за допомогою попередньо натренованої моделі з подальшим озвученням результатів аналізу.
  + Можливість швидкого виклику екстрених служб країни перебування користувача у разі надзвичайної ситуації.
  + Автоматичне визначення мови інтерфейсу застосунку відповідно до мовних налаштувань пристрою користувача.

2.2 Нефункціональні вимоги

* Доступність – застосунок повинен бути кросплатформним і працювати на основних мобільних платформах – iOS та Android.
* Продуктивність – обробка користувацьких запитів (наприклад аналіз об’єктів) повинна займати мінімальну кількість часу.
* Надійність – система повинна стабільно працювати без помилок і аварійних завершень, забезпечуючи коректну роботу.
* Масштабованість – проєктна архітектура повинна забезпечувати можливість інтеграції нового функціоналу без необхідності переписування існуючих модулів.
* Безпека – персональні дані користувача повинні бути захищені відповідно до чинних стандартів інформаційної безпеки.
* Зручність користування – інтерфейс застосунку має бути інтуїтивно зрозумілим та доступним для користувачів з вадами зору, з урахуванням підтримки голосового керування, масштабування тексту та високої контрастності.
* Мінімальні апаратні вимоги – застосунок повинен працювати на більшості сучасних мобільних пристроїв середнього рівня без суттєвих втрат у швидкодії.

3 ОПИС СИСТЕМИ

Система побудована на основі клієнт-серверної архітектури, де мобільний застосунок виступає у ролі клієнта, що взаємодіє із серверною частиною за допомогою HTTP API-запитів. Розробка здійснювалась з використанням сучасного стеку технологій, зокрема Spring Boot для написання основної серверної частини, React Native для створення кросплатформного мобільного клієнта, а також Flask як допоміжного інструменту для реалізації окремого модуля, пов’язаного з обробкою та аналізом зображень.

3.1 Архітектура проєкту

Розроблена архітектура базується на класичній клієнт-серверній моделі, яка забезпечує чіткий розподіл обов’язків між клієнтською та серверною частинами. Основні компоненти системи:

* Клієнтська частина (мобільний застосунок), реалізована за допомогою React Native, забезпечує зручний та інтуїтивно зрозумілий інтерфейсу, а також надає доступ до основного функціоналу шляхом взаємодії з сервером через RESTful API.
* Серверна частина, реалізована за допомогою Spring Boot, відповідає за обробку запитів від клієнта, взаємодію з базою даних і системою кешування, керування обліковими записами користувачів та їхніми даними.
* Модуль мультимодального розпізнавання, реалізований за допомогою Flask, відповідає за обробку зображень та розпізнавання об’єктів за допомогою попередньо натренованої моделі типу “мова-зір” (vision language model) Moondream. Цей модуль відокремлений від основної серверної частини, що забезпечує можливість його масштабування та незалежної підтримки.
* База даних PostgreSQL використовується для зберігання особистих даних користувача, контактів важливих осіб, номерів екстрених служб у різних країнах, текстових ресурсів відповідно до обраної мови.
* Розподілене сховище Redis використовується для кешування частих запитів, а також для зберігання тимчасових даних.

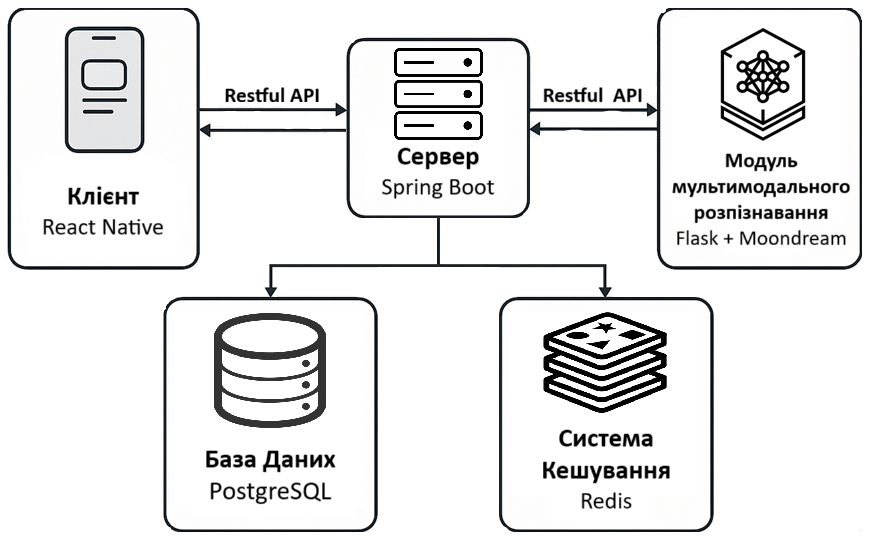


Рисунок 3.1 – Схема архітектури проєкту

3.2 Використані технології

Для побудови системи були використані сучасні технології, підібрані відповідно до функціональних потреб мобільного застосунку та вимог користувачів. Нижче наведено короткий опис цих інструментів:

* TypeScript з використанням React Native та Expo – TypeScript, як мова зі статичною типізацією, забезпечує надійність, можливість виявлення помилок на ранніх етапах розробки та дозволяє писати більш масштабований і легкий у підтримці код. React Native, як популярний програмний каркас для швидкої реалізації мобільного застосунку, надає можливість утримувати єдину кодову базу для iOS та Android систем. Платформа Expo, яка розширює функціональні можливості React Native, забезпечує доступ до широкого набору нативних інструментів для роботи з камерою, геолокацією, аудіо, push-сповіщеннями, розпізнаванням жестів та іншими функціями мобільного пристрою.
* Java з використанням Spring Boot – Java є строго типізованою об’єктно-орієнтовною мовою програмування для написання надійних, масштабованих і високопродуктивних рішень - від десктопних і мобільних застосунків до консольних програм. Основною філософією Java є можливість написати код, який працює однаково на всіх платформах (“write once – run everywhere”) завдяки Java Virtual Machine (JVM). Ця мова чудово підходить для розробки серверної частини застосунку, а програмний каркас Spring Boot лише розширює можливості та пришвидшує процес розробки, надаючи готові рішення для реалізації REST API, конфігурації безпеки та роботи з базами даних.
* Python з використанням Flask – Python є інтерпретованою динамічно типізованою мовою, яка вирізняється простотою написання коду та зрозумілим синтаксисом. Завдяки великій кількості бібліотек і готових рішень, ця мова широко застосовується для розробки вебзастосунків, аналізу даних, автоматизації рішень та створення систем машинного навчання. Програмний каркас Flask надає зручний інструментарій для побудови сервера для обробки HTTP-запитів, що дозволяє швидко розробляти та розгортати API з мінімальними затратами на конфігурацію.
* PostgreSQL – це потужна об’єктно-реляційна система управління базами даних з відкритим початковим кодом, яка призначена для зберігання, обробки та керування великими обсягами даних. Вона розроблена з акцентом на розширюваність і сумісність зі стандартами SQL.
* Redis – це високопродуктивне сховище даних в оперативній пам’яті з відкритим початковим кодом, яке працює за принципом “ключ-значення”, та використовується для кешування даних, а також для забезпечення комунікації між компонентами розподілених систем.
* Moondream – це модель типу “мова-зір” з відкритим початковим кодом, яка надає можливості обробки та аналізу зображень на основі текстових запитів, а також для виявлення об’єктів. Її головною перевагою є баланс між ефективністю й точністю результатів при мінімальному споживанні ресурсів.

Для забезпечення стабільності програмного забезпечення під час розробки, а також для кращої підтримки та масштабованості були використані такі технології:

* Git – це система контролю версій, що дозволяє ефективно керувати змінами в коді та зберігати історію розробки, а також є незамінним інструментом для командної роботи.
* Swagger (OpenApi) – це інструмент для опису та тестування REST API, який полегшує взаємодію між клієнтською та серверною частина застосунку, забезпечуючи зрозумілу та інтерактивну документацію.
* Liquibase – це інструмент для керування версіями бази даних, який дозволяє автоматизувати процеси створення, оновлення та відстеження змін у схемі бази даних.
* Docker – це платформа для контейнеризації програмного забезпечення, яка дозволяє створити уніфіковане середовище для розробки, тестування та розгортання. За допомогою docker-compose файлу можна легко керувати мультиконтейнерними застосунками, налаштовуючи всі компоненти системи та мережу між ними в одному конфігураційному файлі, що значно спрощує адміністрування складних середовищ.

Зображення, що містить текст, знімок екрана, Шрифт, число

Вміст, створений ШІ, може бути неправильним.Рисунок 3.2 – Контейнеризовані компоненти застосунку

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Серверна частина

4.1.1 Модуль авторизації та автентифікації

Для забезпечення захисту користувацьких даних та розмежування функціоналу між різними типами користувачів (особами з порушеннями зору та контактними особами) реалізовано модуль авторизації та автентифікації на основі JWT (JSON Web Token). Даний модуль відповідає за перевірку особистості користувача, генерацію токенів доступу та їх перевірку при кожному запиті.

Для автентифікації використовується поєднання електронної пошти та паролю з початковою верифікацією електронної пошти за допомогою листа, що надсилається за допомогою JavaMailSender API. Окрім того, передбачено можливість автентифікації через обліковий запис Google. Під час первинної автентифікації користувач повинен вказати свою роль для коректного розподілу прав доступу та функціоналу.

Зображення, що містить текст, число, Шрифт, ряд

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.1 – Документація API модуля, створена за допомогою Swagger

4.1.2 Модуль опрацювання локації користувача

Для перетворених географічних координат місцезнаходження користувача, отриманих за допомогою нативних можливостей мобільного пристрою, у текст, зрозумілий для сприйняття людиною (що включає назву країни, населений пункт та адресу), реалізовано відповідний модуль. Отримані дані використовуються у двох основних напрямках:

а) під час формування повідомлення в модулі для надсилання сповіщення довіреним особам;

б) для визначення країни з метою подальшого вибору відповідних номерів екстрених служб.

Для здійснення зворотного геокодування (перетворення координат у структурований JSON-об’єкт із необхідною інформацією) використовується OpenStreetMap API.

Зображення, що містить текст, знімок екрана, програмне забезпечення, число

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.2 – Приклад відповіді від OpenStreetMap API

4.1.3 Модуль виклику екстрених служб

Для отримання номерів екстрених служб на основі країни, в якій перебуває користувач, реалізовано відповідний модуль. На початковому етапі реалізації використовувався Emergency Numbers API, однак у процесі тестування було виявлено розбіжності між даними, що поверталися цим API, та реальними номерами служб у деяких країнах. Тому було прийнято рішення сформувати власний перелік актуальних номерів та зберігати його у базі даних застосунку. На поточному етапі реалізація охоплює Україну та країни-сусідів, однак структура модуля дозволяє легко розширити цей перелік у майбутньому.

Зображення, що містить текст, знімок екрана, програмне забезпечення, число

Вміст, створений ШІ, може бути неправильним.

Рисунок 4.3 – Приклад відповіді з номерами екстрених служб

4.1.4 Модуль надсилання сповіщення довіреним особам

Для забезпечення можливості користувачам із вадами зору швидко надіслати сповіщення довіреним особам реалізовано відповідний модуль. Передусім довіреній особі надсилається угода, яку вона повинна підтвердити, щоб мати змогу отримувати сповіщення від користувача з вадами зору. Після підтвердження угоди користувач отримує змогу надсилати повідомлення. Він може обрати один із заздалегідь визначених типів повідомлень, що класифікуються за такими категоріями:

а) Запити про допомогу (Help/Assistance Requests):

* NEED\_ASSISTANCE – Потребую допомоги.
* FEELING\_UNSAFE – Почуваюсь в небезпеці.

б) Інформаційні повідомлення (Informational Messages):

* LOST – Загубився.
* CURRENT\_LOCATION – Моє поточне місцезнаходження.

в) Екстрені повідомлення (Emergency Messages):

* INJURED\_CALL\_FOR\_HELP – Травмований, потрібна допомога.
* NEED\_MEDICAL\_ATTENTION – Потребую медичної допомоги.

Ці повідомлення реалізовано у вигляді переліку констант (enum), що дозволяє спростити їхню обробку на серверній стороні та забезпечити швидкий вибір потрібного варіанту повідомлення на клієнтській стороні.



Рисунок 4.4 – Документація API модуля, створена за допомогою Swagger

4.1.5 Модуль для забезпечення багатомовності

Для забезпечення багатомовності було використано готові рішення програмного каркасу Spring Boot, зокрема клас MessageSource, який відповідає за завантаження локалізованих повідомлень за відповідними ключами. Також було налаштовано розташування локалізованих повідомлень, які зберігаються у файлах з розширенням .properties у директорії resources/messages. Було створено окремі файли для української та англійської мов.

Під час реалізації підтримки української мови виникла проблема з відображенням кириличних символів. Це пов’язано з особливостями кодування файлів .properties, які за замовчуванням використовують кодування ISO-8859-1. Для вирішення цієї проблеми кириличні символи було представлено у вигляді Unicode Escape Sequence (наприклад, \u041f\u0440\u0438\u0432\u0456\u0442 замість "Привіт"), що забезпечило їхнє коректне відображення у застосунку.

Для визначення мови використовується заголовок Accept-Language протоколу HTTP. Якщо заголовок відсутній або мова не підтримується системою, за замовчуванням обирається англійська. Окрім цього, було реалізовано окремий клас TextResource, який забезпечує зручний програмний інтерфейс до локалізованих повідомлень за ключами відповідно до поточної мови.

4.1.6 Модуль для обробки винятків

Для централізованої обробки винятків у серверній частині було реалізовано клас GlobalExceptionHandler, який використовує можливості програмного каркасу Spring Boot. Це дало змогу зосередити логіку обробки винятків в одному місці та повертати користувачеві інформативну відповідь з повідомленням та відповідним HTTP статус-кодом.

Окрім того, було створено допоміжний клас ExceptionUtils, методи якого генерують відповідні винятки з локалізованими повідомленнями, що надалі обробляються GlobalExceptionHandler. У межах архітектури серверної частини це єдине допустиме місце для генерації винятків – використання throw у довільних місцях коду не заохочується.

4.1.7 Контролер для мультимодального розпізнавання

Модуль мультимодального розпізнавання було реалізовано з використанням Python. Водночас клієнтська частина взаємодіє з цим модулем через серверну частину, написану на Java. За допомогою класу WebClient, який є частиною модуля Spring WebFlux, користувацький запит із зображенням (у форматі Base64 або Multipart-файлу) перенаправляється до Python-сервісу з відповідним заголовком для визначення мови відповіді.

4.2 Модуль мультимодального розпізнавання

Модуль мультимодального розпізнавання реалізує дві кінцеві точки REST API для аналізу зображень: один приймає зображення у форматі Base64, інший — у вигляді файлу за допомогою multipart/form-data.

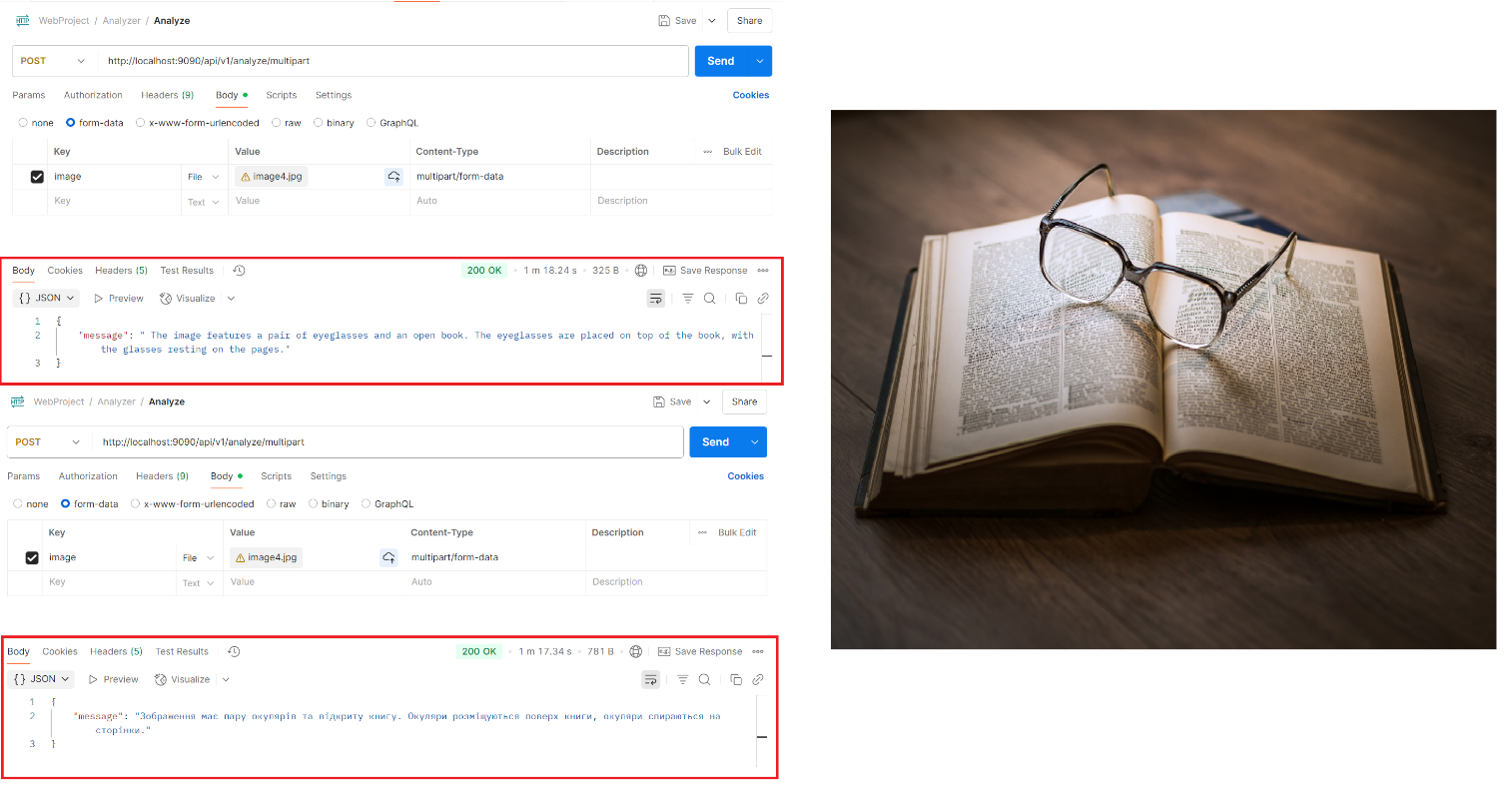
Обробка зображень здійснюється за допомогою моделі Moondream, яка містить 2 мільярди параметрів. Результат аналізу повертається англійською мовою і, за потреби, його можна перекласти на іншу мову, вказану у відповідному заголовку HTTP-запиту. У такому випадку переклад здійснюється автоматично за допомогою бібліотеки Deep Translator, що використовує Google Translate API.

Рисунок 4.5 – Аналіз зображення №1: англійський та український переклад

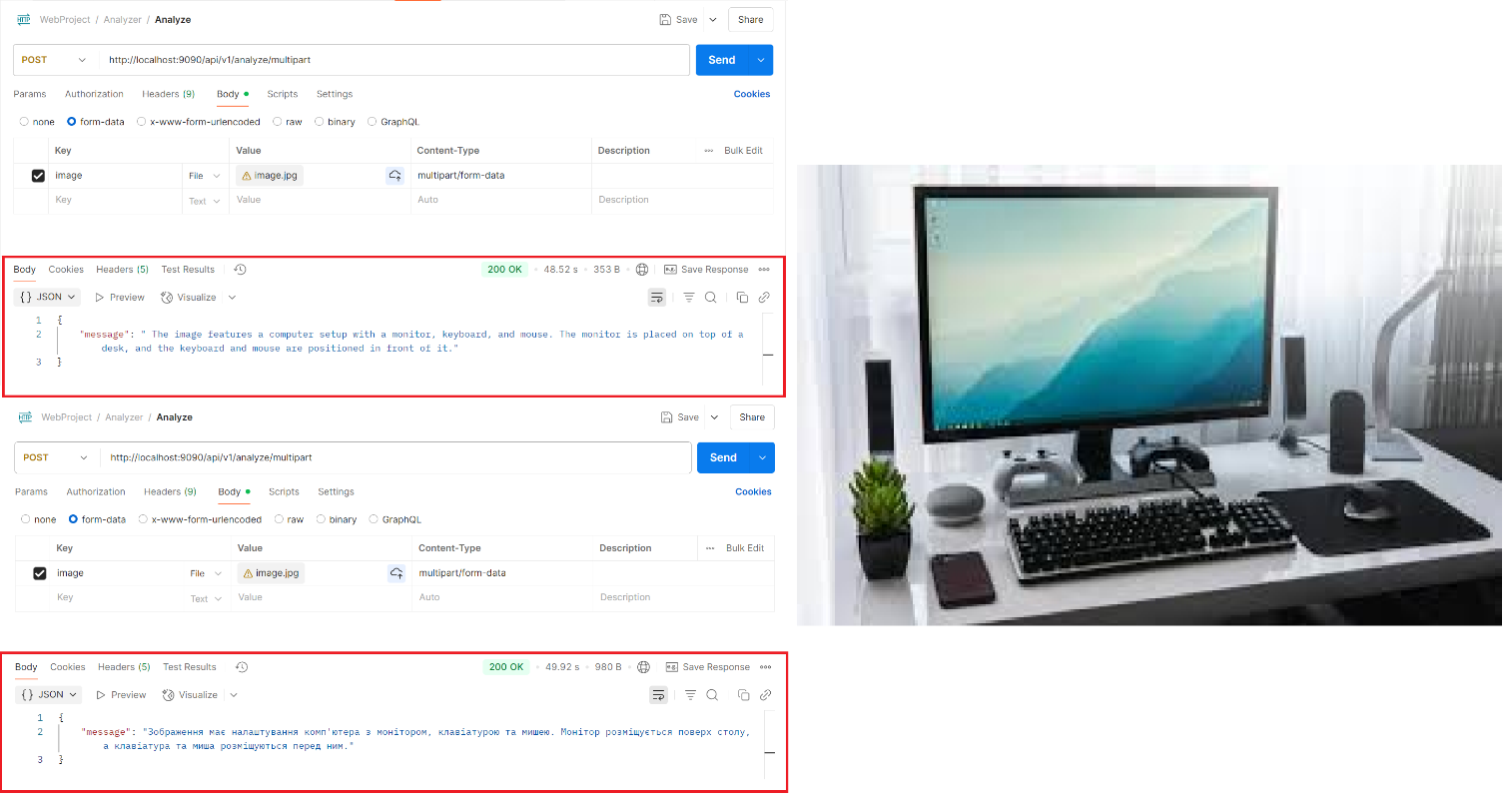


Рисунок 4.6 – Аналіз зображення №2: англійський та український переклад

4.3 Клієнтська частина

4.3.1 Сервіс для надсилання запитів до серверу

Для комунікації із серверною частиною використовується бібліотека axios. Також оголошено єдиний екземпляр (axios instance), який надає можливості:

* оголошувати базову URL-адресу для всіх запитів;
* автоматично додавати HTTP-заголовки, зокрема Bearer Token для авторизації чи Accept-Language для локалізації відповіді;
* централізовано обробляти помилки за допомогою перехоплювачів (interceptors).

Axios спрощує надсилання HTTP-запитів, а також стандартизує взаємодію із серверною частиною.

4.3.2 Модуль голосової взаємодії з користувачем:

Для розробки цього модуля в основному були використані дві бібліотеки: expo-av та expo-speech. Модуль голосової взаємодії виконує такі функції:

а) Відтворення заздалегідь записаних аудіофайлів – для ознайомлення користувача з функціональністю кожного екрана були підготовлені аудіофайли українською та англійською мовами, які відтворюються за допомогою бібліотеки expo-av. Відповідний файл автоматично програється під час першого відвідування екрана або при натисканні кнопки “Пояснення”.

б) Озвучення динамічного тексту – для озвучення результатів запитів або пояснення окремих елементів інтерфейсу використовується бібліотека expo-speech, яка застосовує нативні можливості iOS та Android. Для забезпечення єдиного підходу перетворення тексту на звук була реалізована функція speak, яка стандартизує параметри голосу, швидкість і тембр відтворення.

4.3.3 Модуль тактильних відкликів

Для реалізації тактильних ефектів було використано бібліотеку expo-haptics, яка використовує нативні можливості пристрою для створення вібрацій, які можуть мати різну інтенсивність та тривалість. У межах модуля розроблено набір допоміжних функцій, які абстрагують використання цієї бібліотеки та дозволяють легко інтегрувати тактильні відклики в будь-яку частину коду.

Ключовим елементом модуля є власний обробник подій useDoublePress, який реалізує механізм подвійного натискання. При першому натисканні обробник ініціює тактильний відклик (і, за потреби, звуковий супровід), щоб повідомити користувача про активність елемента. При повторному натисканні протягом певного інтервалу часу виконується основна дія.

4.3.4 Модуль для опрацювання локації користувача

У межах створеної системи для багатьох сценаріїв важливо знати місцезнаходження користувача. Для отримання географічних координат пристрою використовується бібліотека expo-location, яка надає зручний інтерфейс для доступу до геолокаційних сервісів.

Також реалізовано функцію getCurrentLocation, яка запитує дозвіл у користувача на обробку даних про його місцезнаходження, а після цього повертає результат у форматі широти та довготи. Ця функція інкапсулює всю необхідну логіку та обробляє винятки, що забезпечує її зручне та безпечне використання в інших частинах коду.

4.3.5 Модуль для аналізу об’єктів

Застосунок використовує бібліотеку expo-camera та серверні можливості для аналізу об’єктів за допомогою камери мобільного пристрою.

Після захоплення зображення воно проходить обробку для зменшення розмитості та покращення якості. Оброблене зображення надсилається на сервер, де виконується розпізнавання об’єктів. У результаті аналізу сервер формує текстовий опис виявлених об’єктів.

Цей опис повертається до клієнтської частини застосунку, де він перетворюється у звукове повідомлення. Таким чином, користувач отримує голосовий опис навколишніх об’єктів.

4.3.6 Модуль для забезпечення багатомовності

Для реалізації підтримки кількох мов у застосунку було створено модуль багатомовності, який забезпечує динамічне відображення інтерфейсу відповідною мовою. Це було досягнуто завдяки використанню бібліотеки expo-localization, яка зчитує налаштування мови на пристрої користувача. На даному етапі реалізовано підтримку української та англійської мов.

Для локалізації текстових елементів застосунку, навіть попри їхню обмежену кількість, використано бібліотеку i18next. Переклади текстових елементів зберігаються в окремих JSON-файлах для кожної мови, що дозволяє легко розширювати перелік підтримуваних мов у майбутньому.

Крім того, кожній іконці інтерфейсу було присвоєно локалізований опис, що дає змогу програмам зчитування з екрана (screen readers) правильно озвучувати їхнє призначення для користувачів із вадами зору.

4.3.6 Навігація між екранами

Для організації навігації у межах застосунку використано бібліотеку react-navigation/native. Як основний тип навігатора було обрано TabNavigator, що дозволяє зручно перемикатися між основними екранами через нижнє таб-меню.

З метою покращення користувацького досвіду також було реалізовано підтримку жестів для перемикання між екранами. Для цього використано бібліотеку react-native-gesture-handler, яка дозволяє зчитувати жести проведення пальцем, що обробляються в застосунку та забезпечують перехід на наступний або попередній екран.

5 ТЕСТУВАННЯ СИСТЕМИ

Для забезпечення коректної роботи застосунку, відповідності функціональним вимогам і стабільності всієї системи в процесі розробки було проведено багатоетапне ручне тестування її ключових компонентів. Зокрема, було зроблено:

* Функціональне тестування REST API за допомогою Postman. Було перевірено коректність відповідей сервера, обробку помилок (наприклад, невалідні дані, відсутність прав доступу), відповідність структури запитів і відповідей, а також роботу механізмів автентифікації та авторизації.
* Тестування клієнтської частини, яке виконувалося вручну на різних пристроях та екранах — як на емуляторах, так і на реальних смартфонах. Перевірялися основні сценарії використання застосунку, навігація між екранами та взаємодія із сервером.
* Тестування модуля мультимодального розпізнавання проводилося за допомогою Postman з використанням тестових вхідних даних (зображень). Було перевірено точність і стабільність розпізнавання об’єктів, обробку неочікуваних або неповних даних, а також передачу результатів на сервер.

У результаті тестування вдалося виявити й усунути помилки, забезпечити належну взаємодію між модулями та досягти стабільної роботи застосунку на різних платформах. Таким чином, проведене тестування підтвердило відповідність системи технічним завданням та функціональним вимогам.

6 ОГЛЯД РЕЗУЛЬТАТІВ

У межах розробки мобільного застосунку було реалізовано три основні екрани, що забезпечують комплексний функціонал для допомоги людям із порушеннями зору.

1. Екран визначення локації користувача – забезпечує точне визначення географічного положення пристрою з використанням вбудованих GPS-модулів. Це дозволяє користувачу отримувати актуальну інформацію про своє місцезнаходження.

Зображення, що містить текст, Мобільний телефон, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.

Рисунок 6.1 – приклад використання екрану для визначення локації

1. Екран аналізу об’єктів – дозволяє користувачу швидко отримувати інформацію про навколишні предмети за допомогою камери мобільного пристрою. Зроблене фото автоматично обробляється, і користувач отримує голосовий опис виявлених об’єктів.

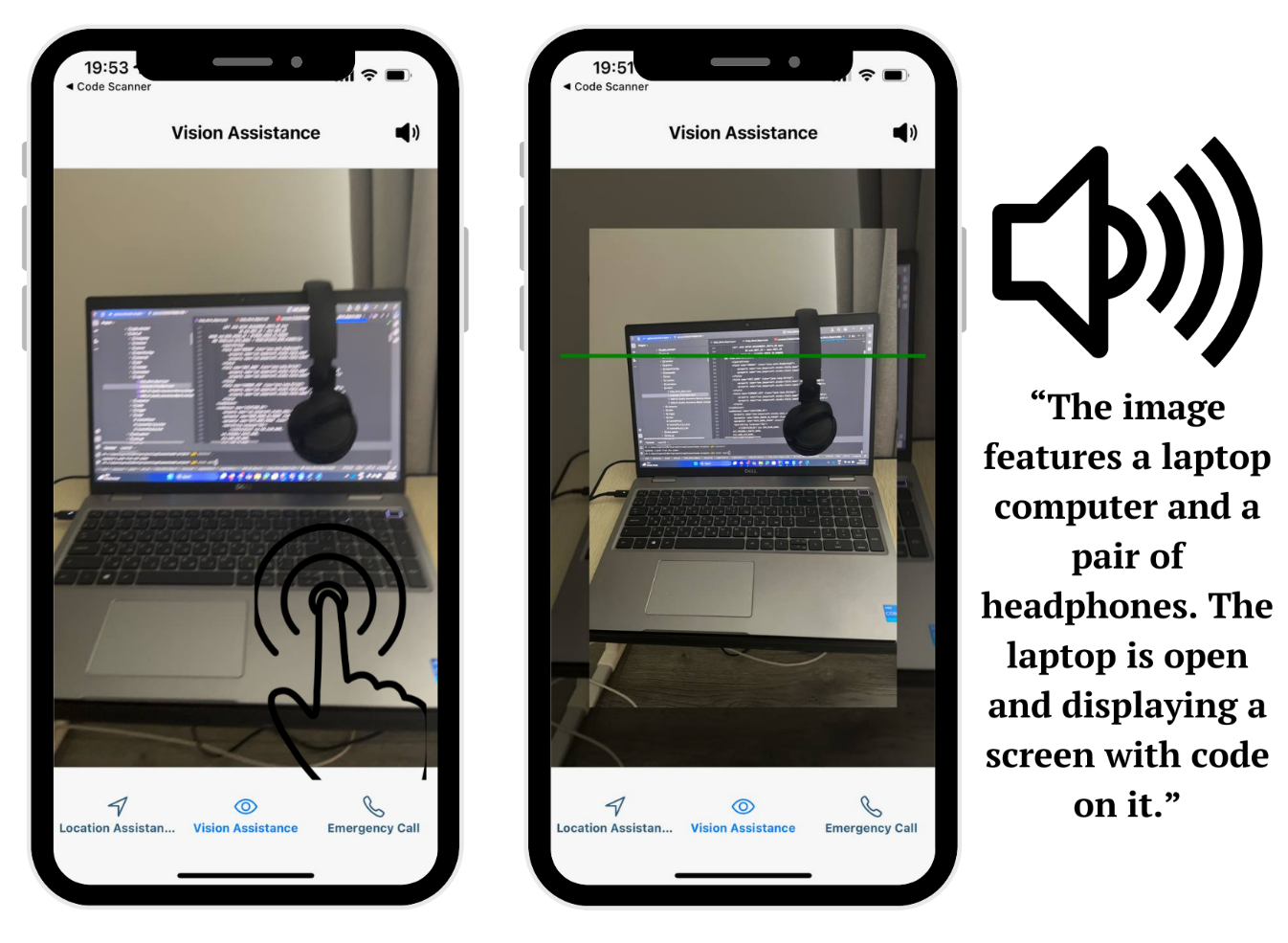


Рисунок 6.2 – приклад використання екрану для аналізу об’єктів

1. Екран виклику екстрених служб – надає можливість швидкого виклику допомоги у критичних ситуаціях залежно від країни, адже номери екстрених служб завантажуються динамічно.



Рисунок 6.3 – приклад використання екрану виклику екстрених служб

Окрім того, на серверній частині було реалізовано важливі функції безпеки та підтримки користувача:

* Автентифікація та авторизація, що забезпечують захищений доступ до сервісів застосунку.
* Механізм надсилання екстрених сповіщень довіреним особам, який дозволяє оперативно інформувати близьких про надзвичайні ситуації.

Варто зазначити, що наразі функціонал надсилання екстрених сповіщень та частина системи автентифікації і авторизації реалізовані на сервері, але ще не інтегровані у клієнтську частину застосунку. Це планується зробити у наступних ітераціях розробки.

Особливу увагу було приділено зручності інтерфейсу: реалізовано підтримку подвійного дотику (double press) для взаємодії з екраном, голосовий супровід усього ключового функціоналу, а також тактильні відклики при взаємодії з об’єктами інтерфейсу, що суттєво підвищує доступність і комфорт користування застосунком для людей із порушеннями зору.

Також було реалізовано функціонал підтримки багатомовності, який на поточному етапі охоплює українську та англійську мови. Мовні налаштування автоматично зчитуються з пристрою користувача, що дає змогу адаптувати інтерфейс, зокрема текстові елементи та голосові підказки, під обрану мову без додаткових налаштувань з боку користувача.

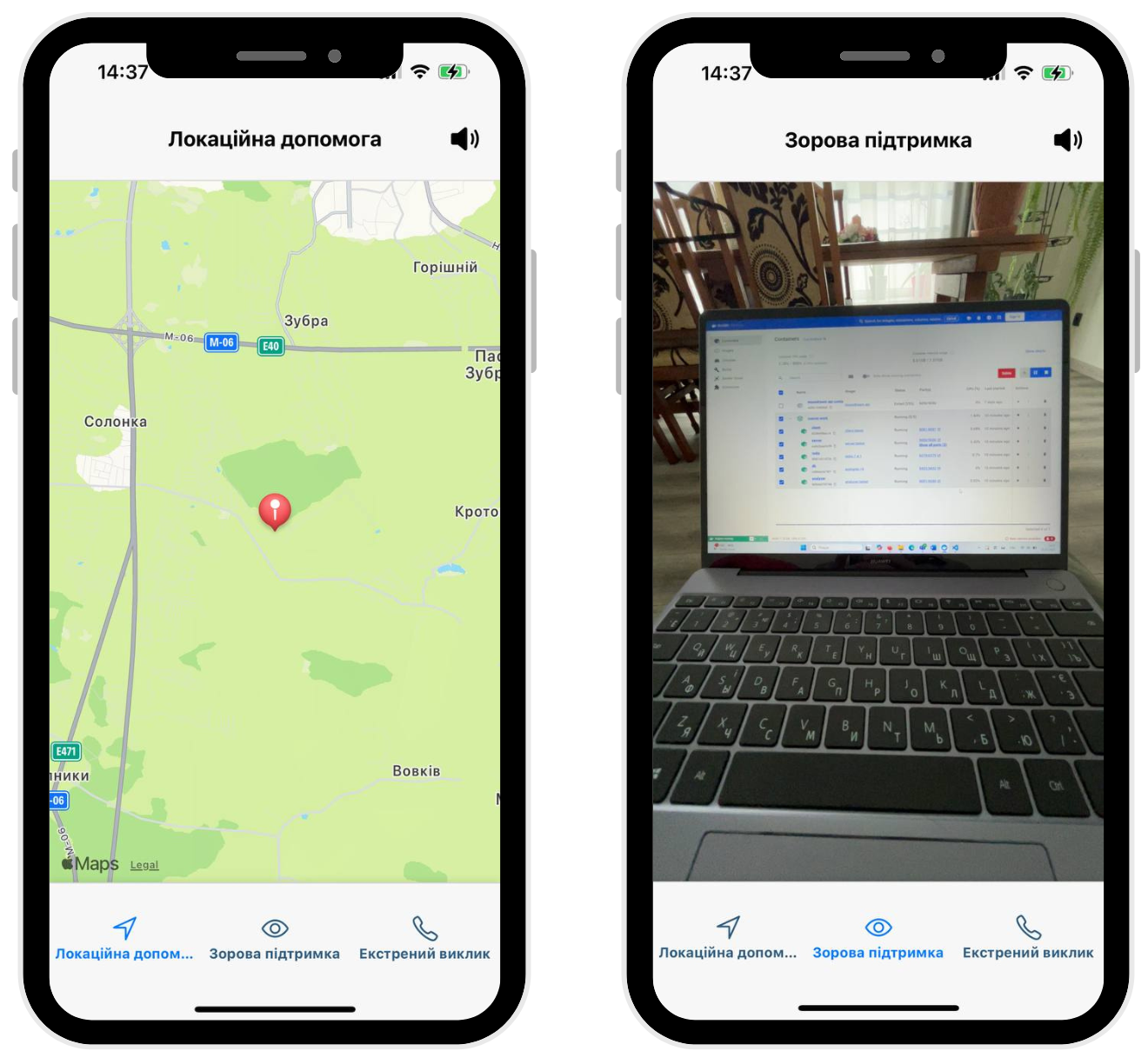


Рисунок 6.4 – користувацький інтерфейс українською мовою

ВИСНОВОК

У ході виконання курсової роботи було розроблено мобільний застосунок, призначений для допомоги людям з вадами зору. Програмний продукт включає базові функції визначення місцезнаходження, аналізу навколишніх об’єктів та можливість швидкого виклику екстрених служб.

На даному етапі застосунок демонструє основний функціонал, який може бути розширений у подальшому. Можливі напрями розвитку включають покращення системи навігації, розширення бази даних екстрених служб та додавання нових функцій для підвищення зручності користування.

Робота продемонструвала практичну реалізацію технологічних рішень для підтримки людей з особливими потребами. У процесі розробки було проведено кілька зустрічей із представниками Львівської обласної організації Українського товариства сліпих (ЛОО УТОС) за сприяння голови правління, що дозволило краще врахувати реальні потреби цільової аудиторії. Отримані результати можуть стати основою для подальших досліджень у сфері доступних технологій.

СПИСОК ДЖЕРЕЛ

1. Офіційний веб-сайт ЛОО УТОС <http://cputos.org.ua/oseredky-utos/lvivska-oblasna-orhanizatsiya-utos/>
2. Офіційна документація Spring Boot <https://docs.spring.io/spring-boot/index.html>
3. Офіційна документація Flask <https://flask.palletsprojects.com/en/stable/>
4. Офіційна документація React Native <https://reactnative.dev/docs/getting-started>
5. Офіційна документація Expo <https://docs.expo.dev/>
6. Офіційна документація PostgreSQL <https://www.postgresql.org/docs/>
7. Офіційна документація Redis <https://redis.io/docs/latest/>
8. Офіційний веб-сайт Moondream <https://moondream.ai/>
9. Код розробленого застосунку https://github.com/nazariusPr/CourseWork