Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для подорожей визначними місцями. Мобільний додаток.
(тема)

Виконала:
студентка 4 курсу, групи ПЗПІ-20-7

Вельма А.О
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Онищенко К.Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Вельмі Анастасії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для подорожей визначними місцями. Мобільний додаток.

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи Розробити застосунок для організації подорожей визначними місцями за допомогою Flutter.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	15.04.2024	*виконано*
2	Формування вимог до ПЗ	20.04.2024	*виконано*
3	Проектування ПЗ	22.04.2024	*виконано*
4	Розробка ПЗ	20.05.2024	*виконано*
5	Тестування ПЗ	22.05.2024	*виконано*
6	Оформлення пояснювальної записки	03.06.2024	*виконано*
7	Перевірка роботи на антиплагіат та проходження нормоконтролю	04.06.2024	*виконано*
8	Оцінка роботи рецензентом, отримання відгуку від керівника кваліфікаційної роботи	14.06.2024	*виконано*
9	Попередій захист роботи	14.06.2024	*виконано*
10	Здача роботи до електронного архіву, допуск роботи до захисту завідувачем кафедри	15.06.2024	*виконано*
11	Захист кваліфікаційної роботи	18.06.2024	

Дата видачі завдання 08 квітня 2024р.

Студентка _____ Вельма А.О.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Онищенко К.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 73 стор., 38 рис., 3 табл., 7 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, FLUTTER, ПОДОРОЖІ, ПЛАНУВАННЯ І ОРГАНІЗАЦІЯ, LANDMARK RECOGNATION

Об’єкт розробки – програмна система для подорожей визначними місцями.

Мета розробки – створення мобільного додатку для подорожей визначними місцями за допомогою фреймворку Flutter. Це включає забезпечення користувачів зручним та інтуїтивно зрозумілим інструментом для планування і організації своїх подорожей. Використання фреймворку Flutter дозволить досягти високої продуктивності та плавності роботи застосунку.

Метод рішення – середовище розробки VS code, мова програмування Dart, фреймворк Flutter.

У результаті розробки створено мобільний застосунок для подорожей визначними місцями за допомогою фреймворку Flutter, що допомагає краще планувати та організовувати подорожі.

MOBILE APP, FLUTTER, TRAVEL, PLANNING AND ORGANIZATION, LANDMARK RECOGNITION

The object of development is a software system for sightseeing travels.

The goal of the development is to create a mobile application for sightseeing travels using the Flutter framework. This includes providing users with a convenient and intuitive tool for planning and organizing their travels. The use of the Flutter framework will achieve high performance and smooth operation of the application.

Solution method – VS code development environment, Dart programming language, Flutter framework.

The development resulted in a sightseeing mobile app using the Flutter framework that helps you better plan and organize your trips.

Я, Вельма Анастасія Олександрівна, студентка гр.ПЗПІ-20-7, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для подорожей визначними місцями. Мобільний застосунок», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному світі, коли подорожі стають все більш популярними і доступними, користувачі шукають зручні та ефективні способи планування та організації своїх подорожей. Часто вони стикаються з проблемою знаходження цікавих та визначних місць для відвідування, а також з необхідністю зберігання важливої інформації для подорожі. Проектування та визначення основних вимог мобільного застосунку, який надає можливість швидко та якісно організовувати та планувати подорожі, є основною темою цієї кваліфікаційної роботи. Основне завдання полягає у створенні інструменту, який забезпечить зручність та якість процесу планування подорожей для користувачів. Метою роботи є реалізація програмної системи, яка передбачає створення мобільного застосунку, що дозволить користувачам з легкістю знаходити цікаві місця для відвідування та зберігати корисну інформацію про подорожі.

Для розробки запропонованого продукту було обрано мову програмування Dart та фреймворк Flutter, віддаючи перевагу їхній комбінації через їхню ефективність та можливість швидкого розвитку. В якості середовища розробки було використано VS Code, що відоме своєю зручністю та розширюваністю для роботи з Dart і Flutter.

Застосунок має полегшити процес планування місць, які користувач хоче відвідати, формування бюджету конкретної поїдки, а також збирати усю інформацію з поїздки (фотографії, відвідувані місця тощо) для подальшого використання.

Запропонований продукт стане надійним помічником для мандрівників, допомагаючи їм ефективно планувати свою подорож, знаходити цікаві місця для відвідування та зберігати всю необхідну інформацію в одному зручному місці. Такий застосунок підвищить комфорт та задоволення від подорожей, зекономивши час і зусилля користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний огляд предметної області

В наш час багато людей подорожують світом, відкриваючи нові країни, культури та пригоди. Сфера подорожей та організації поїздок стає все більш популярною в світі. Люди прагнуть знайомитися з культурами інших країн, відвідувати нові визначні місця, а також створювати незабутні спогади. Однак планування подорожі та її організація є складним процесом, що займає багато часу та зусиль [1], тому ефективна організація подорожей стає найважливішим завданням у цьому процесі. Саме тому виникає потреба у зручному інструменті, який би допомагав користувачам ефективно організовувати свої подорожі [2]. Розробка такого мобільного застосунку «Tripster» за допомогою фреймворка Flutter значно спростить процес планування подорожі. Застосунок допоможе мандрівникам забезпечити зручність на кожному етапі їх поїздки.

На ринку вже існують кілька мобільних застосунків, що виконують схожі функції, а саме: TravelSpend, Expedia та TripIt. Вони пропонують широкий спектр функцій, що допомагають організувати подорож з максимальним комфортом і ефективністю. Однак, кожен застосунок має свої переваги і недоліки.

TravelSpend відомий своєю можливістю відстежувати витрати під час подорожі, що дозволяє користувачам керувати своїми фінансами. Однак, його обмеження полягають у відсутності можливості бронювання квитків та готелів, а також у нього є платна підписка. На рисунку 1.1 зображено скріншоти цього застосунку.

Expedia вражає своїм великим вибором готелів, квитків та оренди авто, а також зручним пошуком та порівнянням цін. Проте, він не надає інструментів для планування маршрутів та бюджету, а його інтерфейс є складним та не актуальним для користувачів. Дизайн цього застосунку має проблеми з перевантаженням інформації на екрані, що є нелегким для користувачів під час взаємодії з застосунком. На рисунку 1.2 зображено скріншоти цього застосунку.

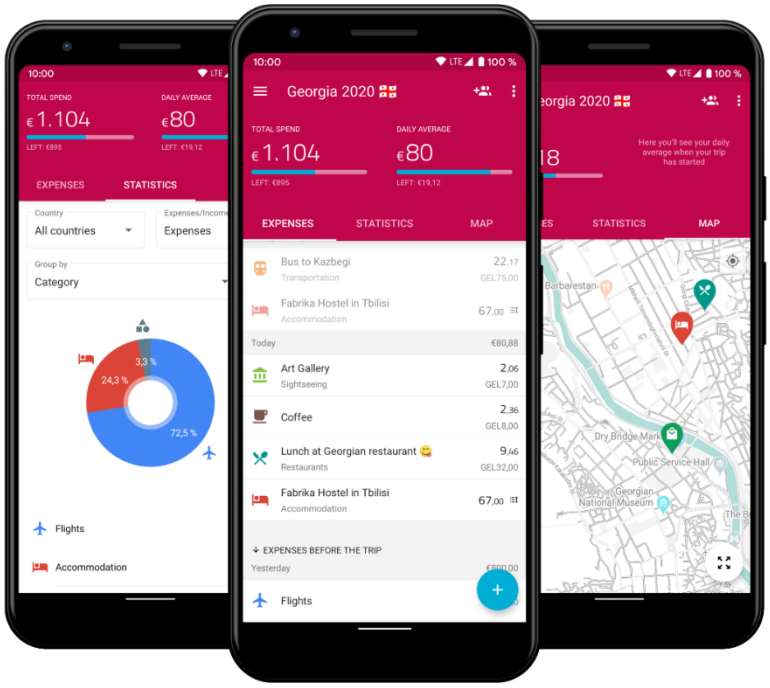


Рисунок 1.1 – Застосунок TravelSpend

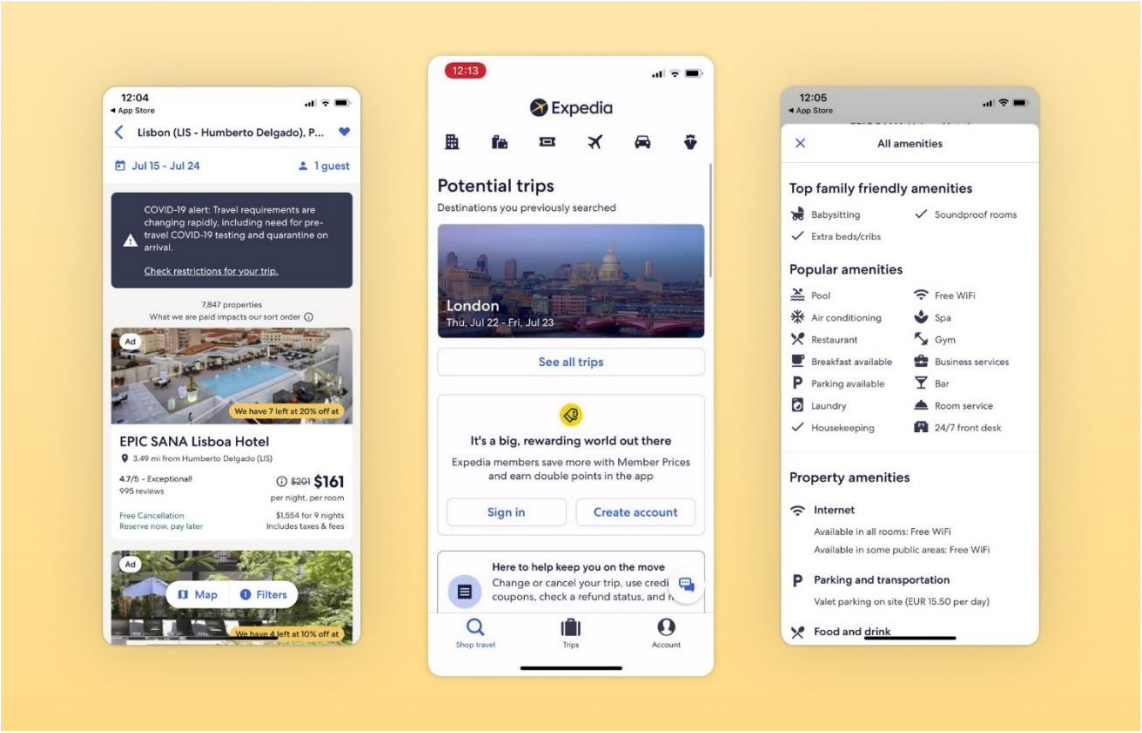


Рисунок 1.2 – Застосунок Expedia

TripIt приваблює тим, що збирає всю інформацію про подорож в одному місці, має офлайн-доступ та автоматичне оновлення інформації. Але він не дозволяє бронювати квитки та готелі, його безкоштовна версія обмежена та

синхронізація з календарем може бути неповною. На рисунку 1.3 зображено логотип цього застосунку.

Рисунок 1.3 – Логотип застосунку TripIt

Загальні плюси мобільних застосунків для планування подорожей включають їх зручність та мобільність, економію часу та доступ до інформації, а також можливість планувати бюджету. Однак, вони також мають загальні недоліки, такі як обмежений функціонал, недосконалі алгоритми, реклама та незручний інтерфейс. Крім того, вони є платними та містять багато недоречної реклами, що є незручним для користувачів. Це все ускладнює для користувачів організацію повноцінних поїздок до визначних місць, тому виникає потреба у розробці програмної системи, яка ефективно буде організовувати планування таких подорожей.

Для того щоб, розробити запропонований мобільний застосунок було обрано кросплатформлену технологію Flutter від Google [3]. Застосунки, розроблені з Flutter, відрізняються високою швидкістю роботи та плавною анімацією [4]. Це робить їх комфортними та простими у використанні. Використання Flutter дозволить створити високоякісний, зручний та доступний для широкого кола користувачів мобільний застосунок.

1.2 Виявлення та вирішення проблем

Після аналізу предметної галузі виявлено декілька проблем, що ускладнюють процес організації поїздки для мандрівників. Перша проблема полягає в тому, що існуючі мобільні застосунків зосереджені лише на окремих аспектах подорожей, таких як бронювання готелів або пошук авіаквитків. Це робить процес планування неефективним, оскільки користувачам доводиться використовувати кілька різних застосунків для організації повноцінної поїздки. Тому розробка запропонованого єдиного мобільного застосунка, який об'єднає всі етапи планування подорожі, від пошуку визначних місць до планування подорожі, а також збереження необхідної інформації, стане важливим кроком у вирішенні цієї проблеми.

Наступна проблема полягає в обмежених можливостях для створення персоналізованих планів подорожей. Більшість існуючих застосунків пропонують лише стандартні маршрути або рекомендації, не даючи користувачам можливості створювати власні плани відповідно до їхніх уподобань. Тому важливо впровадити функціонал для створення персоналізованих планів подорожей, з можливістю вибору визначних місць та заходів, що відповідають інтересам користувачів.

Іншою проблемою є те, що деякі застосунки не надають зручного способу збереження та перегляду спогадів з подорожі. Додавання функцій для зберігання фотографій, нотаток та інших спогадів у мобільний застосунок допоможе користувачам зберігати та переглядати свої враження від подорожі зручним та організованим способом.

Останньою проблемою є те, що мало застосунків пропонують розпізнавання місць за допомогою камери. Застосування технології розпізнавання визначних місць у мобільному застосунку дозволить користувачам отримувати інформацію про місця навколо них просто за допомогою камери свого смартфона.

Запропонована програмна система для подорожей «Tripster» стане ідеальним рішенням для планування та організації поїздок визначними місцями. Вона дозволить мандрівникам створювати особистий план подорожі, переглядати створений бюджет, зберігати враження та розпізнавати місця за допомогою камери.

1.3 Постановка задачі

Для вирішення актуальної проблеми для користувачів, які подорожують, потрібно створити мобільний застосунок «Tripster», що буде забезпечувати ефективну організацію подорожей. Запропонований продукт повинен допомагати користувачу у плануванні подорожі та маршруту, створенні фотоальбомів для збереження спогадів та у розпізнаванні визначних місць за допомогою камери. Також застосунок повинен давати змогу переглядати запланований бюджет, а також переглядати персоналізовані рекомендації. Продукт має дати можливість додавати нотатки до кожного дня у плану подорожі.

Застосунок повинен мати інтуїтивний і зрозумілий інтерфейс для зручного використання. Застосунок буде розроблений за допомогою Flutter, що дозволить забезпечити високу продуктивність та масштабованість. Застосунок повинен використовувати сучасний та привабливий дизайн, заснований на концепціях Material Design, який відзначається чистим та сучасним виглядом. Використання Flutter та Material Design гарантує не лише високу продуктивність застосунку, а й привабливий та дружній інтерфейс, що сприяє зручному взаємодії користувача з застосунком.

Успішне створення цього застосунку дозволить користувачам ефективно планувати свої подорожі, враховуючи їхні індивідуальні уподобання та потреби. Такий застосунок може стати незамінним помічником для всіх любителів подорожей, допомагаючи їм зробити кожну поїздку незабутньою та комфортною.

1.3.1 Цільова аудиторія

Швидкий розвиток сучасних засобів комунікації та доступу до інформації дозволяє користувачам широко використовувати мобільні пристрої для планування та організації своїх подорожей. Враховуючи цей тренд, розробка мобільного застосунку стає важливим кроком у відповідь на потреби сучасного користувача. Такий застосунок не лише забезпечить зручний інструмент для планування подорожей, а й буде відповідати вимогам швидкості, доступності та інтуїтивності, що визначають сучасні мобільні застосунки. Запропонована програмна система для

подорожей визначними місцями призначена для широкого кола користувачів, що подорожують. Мобільний застосунок розрахований на аудиторію, яка також зіткнулась з проблемою організації у власних подорожах, або у яких зовсім немає досвіду у плануванні. Програмна система також спрямована на тих, хто має обмежений час для планування подорожей або шукає швидкі та зручні рішення.

Запропонований продукт не обмежується віковими, професійними або статевими критеріями, тому застосунок розрахований на людей будь-якого віку будь-якої професії, які цікавляться подорожами та прагнуть отримати інтуїтивно-зрозумілий застосунок для планування своїх подорожей. Застосунок призначений для широкого кола користувачів, які мають різний рівень досвіду у плануванні та організації подорожей та зацікавлені швидко та якісно керувати своїми подорожами.

Таким чином, запропонована програмна система відповідає потребам сучасного мандрівника, який потребує зручності та ефективності в плануванні своїх подорожей. Враховуючи швидкий розвиток технологій та зростання популярності мобільних застосунків, система надає можливість доступу до необхідної інформації та інструментів у будь-який час та в будь-якому місці. Це особливо важливо в умовах постійного руху та швидкого темпу життя, коли користувачі потребують зручних інструментів для організації своїх мандрівок без зайвих зусиль. Отже, запропонована програмна система націлена на забезпечення найвищого рівня задоволення потреб та очікувань користувачів у сфері подорожей.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Програмна система містить мобільний застосунок для подорожей визначними місцями, що є основним інструментом для користувача під час його подорожей. Мобільний застосунок є невід'ємною часиною цієї системи, тому що допомагає користувачу організовувати подорожі, а саме: створювати план та переглядати бюджет подорожі, зберігати та створювати галерею фотографій для спогадів, переглядати корисні рекомендації та зберігати усю необхідну інформацію. Необхідно, щоб інтерфейс мобільного застосунку був розроблений, використовуючи передові технології мобільної розробки, що гарантуватимуть швидку роботу та комфортність у використанні.

2.2 Технології та інструменти

Для розробки запропонованого мобільного застосунку для подорожей визначними місцями будуть використовуватись наступні технології:

- Flutter – фреймворк для розробки мобільного застосунку. Використовується для швидкої та ефективної розробки інтерфейсу користувача з використанням гнучкого інструментарію та багаторазових віджетів;
- Material Design – дизайн-мова, що розроблена Google, яка визначає стандарти для розробки зручних та зрозумілих інтерфейсів користувача для мобільних застосунків;
- Landmark Recognition API – технологія розпізнавання місць на зображеннях або в реальному часі;
- HTTP – це один з основних пакетів для виконання HTTP-запитів у Flutter. Він дозволяє легко виконувати GET, POST, PUT, DELETE та інші види запитів, обробляти відповіді сервера та взагалі працювати з HTTP-запитами у зручний спосіб;
- Octo_Image – пакет, який дозволяє ефективно завантажувати та відображати зображення, кешувати їх та забезпечувати багатофункціональність щодо маніпулювання зображеннями;

- Provider – пакет для управління станом, який забезпечує простий та ефективний спосіб передачі та оновлення даних між різними віджетами у застосунку;
- Cubit – пакет для управління станом, що надає зручний та прозорий спосіб керування станом застосунку;
- SharedPreferences – пакет для Flutter, який дозволяє зберігати невеликі об'єми даних (наприклад, налаштування або токени аутентифікації) локально на пристрої користувача.

2.3 Структура та архітектура

У застосунку використовуватиметься трирівнева архітектура. Трирівнева архітектура в Flutter – це підхід до створення застосунків, який розділяє різні аспекти розробки на три рівні: представлення (presentation), логіку застосунку (domain) та доступ до даних (data) [5].

На першому рівні розміщаються всі елементи, які стосуються візуального представлення застосунку, такі як віджети, макети, стани та стилі. Другий рівень – це логіка застосунку, де розміщена уся бізнес-логіка. На цьому рівні відбувається обробка подій, взаємодія з сервером та маршрутизація. І третій рівень відповідає за роботу з базами даних, віддаленими джерелами даних та зберіганням інформації.

Ця архітектура допомагає зберігати код застосунку організованим, полегшує розвиток та підтримку, а також робить застосунок більш масштабованим та зручним у використанні. У структурі проекту з використанням трирівнева архітектури, застосунок розділений на такі частини:

- Presentation layer – у цій частині є такі папки як «cubits» та «provider» для управління станом застосунку, «widgets» та «screens» для відображення створених сторінок;
- Data layer – у цій частині міститься папка «models» що обробляє дані JSON, XML із джерел даних. Також міститься папка «DataSources», що зберігає усю взаємодію з базами даних і API. Тут виконуються всі http-

запити, що обробляють кешовані дані. Це приймає необроблені дані з різних ресурсів і надсилає ці необроблені дані через модель;

– Domain layer – тут зберігаються уся логіка застосунку, що містить в собі логіку обробки даних, правила бізнес-процесів та управління застосунком. Є папка «entities», щоб зберігати дані toJson, fromJson.

2.4 Дизайн та користувацький інтерфейс

Дизайн та інтерфейс є критично важливою складовою будь-якого мобільного застосунку. На відміну від веб-сайтів, де користувачі мають більше простору для навігації, мобільний застосунок повинен забезпечити інтуїтивно зрозумілий і ефективний досвід для користувачів, які часто використовують їх у своїх подорожах. У запропонованому мобільному застосунку повинен мати дизайн, що спрямований на максимальну зручність та зрозумілість для користувачів. Один з принципів, що буде застосовано – це простота та мінімалізм в дизайні для того, щоб уникнути перевантаження інформацією та забезпечити легкий доступ до основних функцій. Усі елементи у застосунку будуть розміщені таким чином, щоб користувачі могли швидко зрозуміти, як ними користуватися без необхідності в додаткових інструкціях. Також використовуватимуться приємні для ока кольори та шрифти, щоб створити дружній та привабливий інтерфейс, який стимулюватиме користувачів повертатися до застосунку. У застосунку буде передбачена можливість зміни теми, що дозволить користувачам налаштувати вигляд застосунку згідно з їхніми власними уподобаннями. Головною перевагою буде те, що дизайн буде адаптований до різних розмірів екранів, щоб забезпечити однаково зручний досвід користувача на різних пристроях. Застосунок міститиме дизайн, який не лише допоможе користувачам з легкістю планувати та керувати своїми подорожами, але й зробить цей процес приємним та захоплюючим.

2.5 Взаємодія з серверною частиною

Мобільний застосунок взаємодіятиме з серверною частиною через API для доступу до різноманітною інформації, а саме: дані про визначні місця, інформація

користувача, плани подорожей, фотографії, а також нотатки. Для виконання HTTP-запитів використовуватимуться пакет http, яка надсилає та отримує дані у форматі JSON. Застосунок також буде здатен ефективно обробляти різні типи помилок, які можуть виникнути під час взаємодії з сервером, і надає користувачам зрозумілі повідомлення про виниклі проблеми. Також для покращення швидкодії та зменшення навантаження на сервер, застосунок використовуватиме механізми кешування для зберігання локальних даних. Для кешування зображень буде використовуватись пакет Octo_image, а для зберігання локальних даних – shared preferences.

2.6 Продуктивність та оптимізація

Одним з основних завданням у реалізації мобільного застосунку – є забезпечення високої продуктивності та оптимізація. Для того, щоб досягнути максимальної швидкості та плавності роботи використовуються оптимізовані віджети. У застосунку використовуватимуться віджети Flutter, такі як ListView.builder та GridView.builder для відображення списків або сіток елементів. Це дозволяє ефективно керувати пам'яттю та ресурсами при відображенні великої кількості елементів. Також для підвищення швидкості завантаження та відображення вмісту, використовуватимуться механізми кешування для зберігання та використання раніше завантажених даних. Для оптимізації завантаження та відображення використовується пакет Flutter, такі як Octo_image, які автоматично кешують та оптимізують завантаження зображень. Під час розробки та випуску застосунку активно використовуватимуться інструменти для тестування та профілювання, такі як Flutter DevTools, щоб виявляти та усувати можливі проблеми з продуктивністю та оптимізувати роботу застосунку. Ці підходи допоможуть забезпечити високу продуктивність та ефективність мобільного застосунку, забезпечуючи при цьому приємний досвід для користувачів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПЗ

3.1 UML проектування ПЗ

Для початку роботи над реалізацією програмної системи було проведено детальний аналіз, де були визначені основні функції користувача мобільного застосунку. На рисунку 4.1 зображено діаграму прецедентів, яка відображає взаємодію між акторами та системою в рамках певного контексту. Вона демонструє функціональність системи шляхом ідентифікації основних дій, які можуть бути виконані акторами. На діаграмі зображено два актора: авторизований та неавторизований користувач. Неавторизований користувач має можливість зареєструватись у системі, а також здійснити авторизацію, якщо вже був зареєстрований. Авторизований користувач вже має доступ до багатьох функціональних можливостей. Користувач може переглянути план подорожі, а також переглядати план бюджету подорожі. Також до плану подорожі, користувач має змогу додати, оновити та видалити нотатки.

Рисунок 3.1 – Use-case діаграма мобільного застосунку (Виконана самостійно)

Користувач має можливість створювати та оновлювати свій профіль. Однією з основних функцій є визначення місця за допомогою камери, а також можливість створити альбомів з фотографій для подорожей. Також система дозволяє користувачу переглядати рекомендації щодо місць та активностей. Створення такої

діаграми дозволяє ілюструвати основні функції і взаємодію між акторами та можливостями системи. Ця діаграма використовується як основа для подальшого проектування, розробки і планування робіт над системою.

Для кращого розуміння процесів було створено діаграму діяльності, що графічно відображає послідовність дій, які відбуваються у системі. Ця діаграма дозволяє візуалізувати, які кроки виконуються і як вони взаємодіють один з одним. Діаграма діяльності зображена на рисунку 4.2.

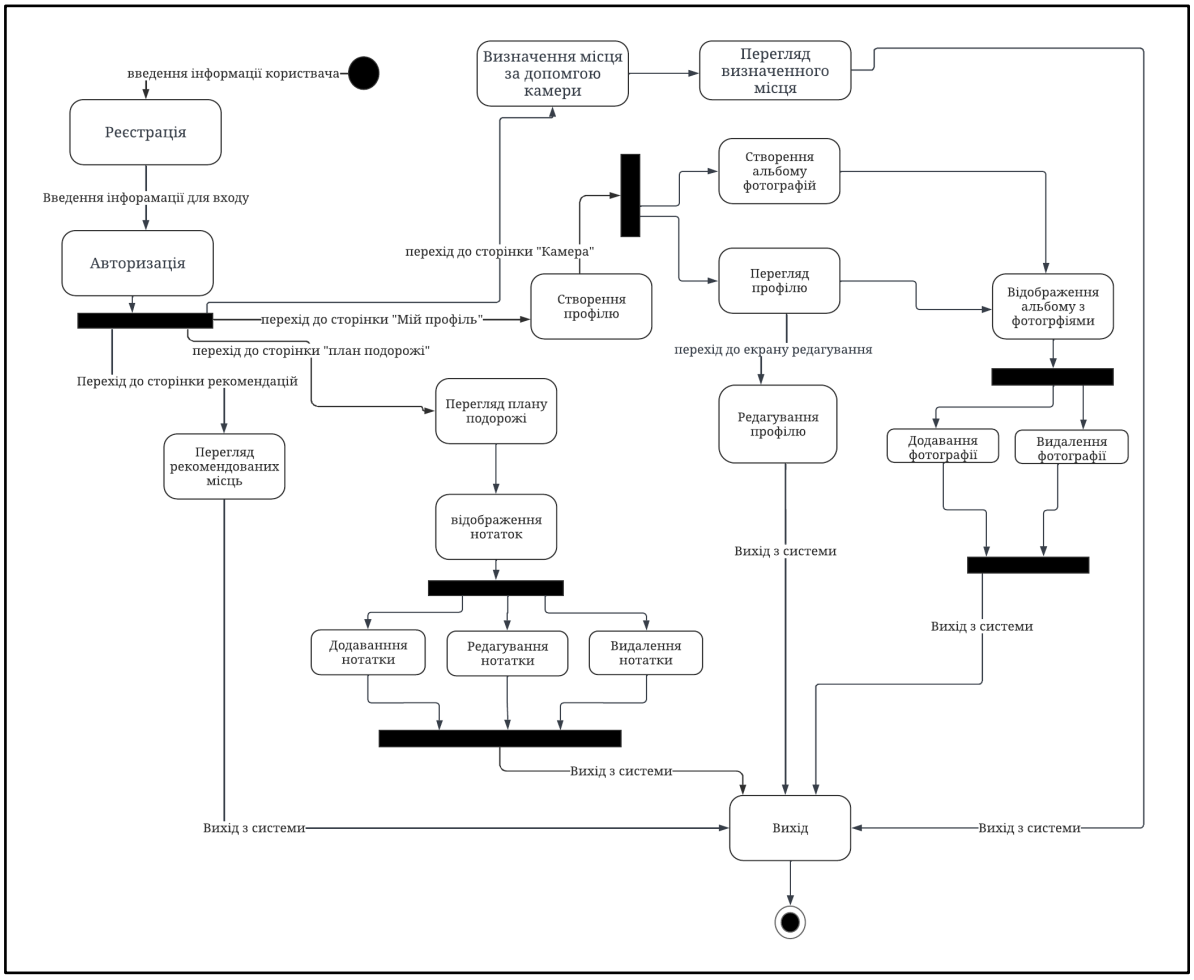


Рисунок 3.2 – Діаграма діяльності мобільного застосунку (Виконана самостійно)

Діаграма починається з точки входу, що є початком взаємодії користувача з мобільним застосунком. Далі користувач здійснює реєстрацію, а згодом авторизацію. Наступним кроком, він має можливість перейти на інші сторінки програми, а саме: «Мій профіль», «План подорожі», «Рекомендовані місця» та «Камера». При перегляді сторінки «План подорожі», користувач має змогу додавати, видаляти та редагувати нотатки. Коли він переходить на сторінку «»Мій

профіль», він може редагувати свою інформацію. Також на цій сторінці, він може створювати альбоми з фотографій та зберігати їх. Користувач має можливість перейти на сторінку «Камера» та здійснити визначення місця за допомогою камери у лайф-режимі, або відкрити з галереї. На сторінці «Рекомендовані місця» є можливість переглядати рекомендовані місця та активності для користувача.

Наступним кроком було розроблено діаграму класів. Діаграма класів мобільного застосунку ілюструє відображення основних класів та їх взаємозв’язків. Основними класами є User (Користувач), Place (Місце), Vacation (Відпустка), VacationDay (День відпустки), Note (Нотатка), та Gallery (Галерея).

Клас «User» містить у собі особисту інформацію користувача таку як: ідентифікатор, ім'я, електронну пошту та хеш пароля. Він має зв’язки з іншими класами, такими як «Place» та «Vacation», що означає користувач має персональні рекомендовані місця, а також має інформацію про відпустки.

Клас «Place» призначений для місць, що можна відвідувати користувачу. Він містить атрибути, такі як ідентифікатор, назва, опис, медіа-файли, розташування, рейтинг, години роботи.

Класи «Vacation» та «VacationDay» містять в собі інформацію про відпустку користувача. Клас «Vacation» містить загальні дані про відпустку, такі як назва, країна, міста, місця, дати, дні відпустки, бюджет та галерею. Клас «VacationDay» представляє окремий день відпустки та містить інформацію про відвідані міста, місця, нотатки та бюджет на цей день.

Клас «Note» призначений для відображення нотаток у кожному дні. Він має в собі ідентифікатор, заголовок та сам запис нотатки. Клас «Gallery» представляє галерею зображень, пов'язаних з відпусткою користувача. Він містить атрибути, такі як ідентифікатор, назва, дата та колекцію зображень. На рисунку 4.3 зображена розроблена діаграма класів.

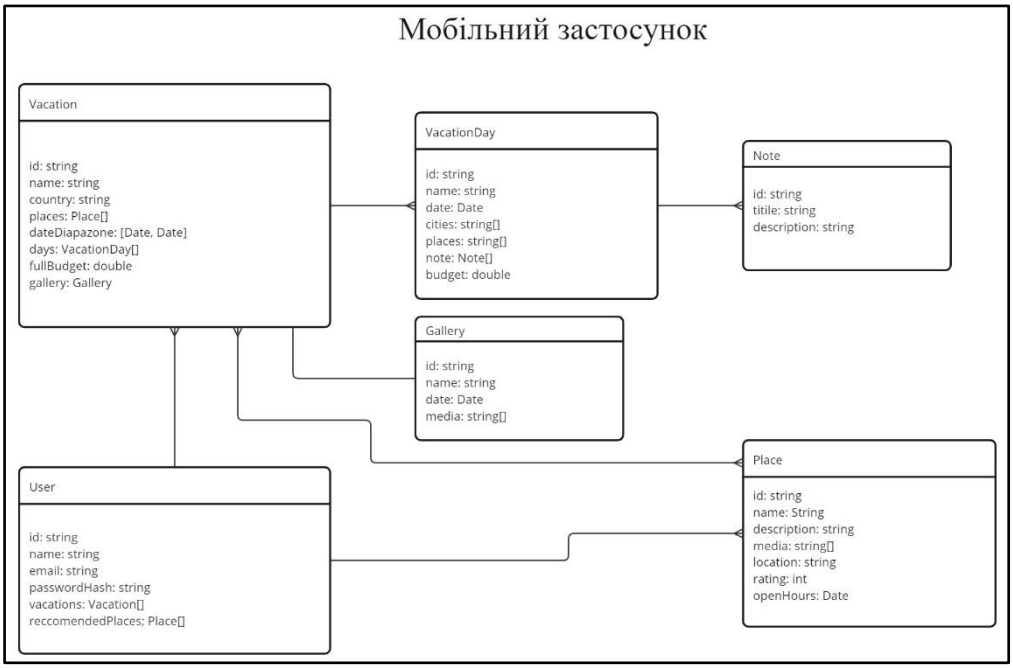


Рисунок 4.3 – Діаграма класів мобільного застосунку (Виконана самостійно)

Розроблена діаграма відображає взаємодію між цими класами та зв’язки між користувачем та функціональністю зі застосунком. Вона допомагає зрозуміти, як дані об’єкти взаємодіють у межах системи та допомагає планувати розробку та реалізацію функціональності застосунку. Ця діаграма є важливим інструментом для аналізу та розуміння архітектури системи та дозволяє ефекти визначати та впроваджувати функціональність в застосунок.

3.2 Проектування архітектури ПЗ

3.2.1 Архітектурний паттерн

Для розробки мобільної частини програмної системи «Tripster» було обрано архітектурний паттерн – Cubit. Він є частиною пакету Bloc (Business Logic Component), який дозволяє організувати бізнес-логіку застосунку та ефективно керувати станом за допомогою потоків даних (streams). [6].

Однією з головних переваг використання Cubit є його простота і зрозумілість. Завдяки чіткій структурі та інтеграції з Flutter, розробники можуть швидко оволодіти цим паттерном і приступити до побудови складних мобільних застосунків. Cubit допомагає спрощувати розробку завдяки тому, що є розділення

бізнес-логіки та представлення даних. Цей паттерн дозволяє створювати окремі компоненти для обробки бізнес-логіки, що підвищує читабельність коду та його підтримку в майбутньому. Керування станом за допомогою потоків даних (streams) дозволяє ефективно реагувати на зміни в застосунку та забезпечує плавну взаємодію з користувачем.

Принципи Cubit базуються на розділенні логіки програми на окремі блоки, що відповідають за конкретні аспекти функціональності. Це сприяє модульності та розширюваності застосунку. Кожен Cubit може мати власні методи для зміни стану і повідомлення про зміни, що дозволяє ефективно керувати взаємодією між компонентами програми.

Використання архітектурного паттерну Cubit дозволяє створити добре структуровану, ефективну та легко розширювану систему, яка забезпечує високу якість та продуктивність мобільного застосунку.

3.3 Проектування структури зберігання, управління даними

3.3.1 Клієнт-серверна архітектура

У мобільному застосунку «Tripster» клієнт-серверна архітектура відображатиме сучасні тенденції у розробці програмного забезпечення, де мобільна частина системи взаємодіє з сервером через API для отримання та передачі даних. Серверна частина системи відповідатиме за зберігання та управління різноманітною інформацією, необхідною для функціонування застосунку. У програмній системі «Tripster» серверна частина виконуватиме важливу роль у забезпеченні користувачам доступу до різноманітних функцій та можливостей. Сервер зберігатиме у собі таку інформацію як: особиста інформація користувачів, рекомендовані місця, плани подорожей та нотатки до них, альбоми фотографій та інші дані. Мобільна частина програмної системи «Tripster» відіграє ключову роль у взаємодії з користувачем та забезпеченні зручного та інтуїтивно зрозумілого інтерфейсу. Мобільний застосунок забезпечить широкий функціонал, включаючи перегляд особистої інформації користувача, список рекомендованих місць для відвідування, плани подорожей, створення та редагування нотаток до подорожей,

перегляд альбомів фотографій та багато іншого. Для цього мобільна частина взаємодіє з сервером через API, запитуючи необхідні дані та оброблюючи їх для відображення у відповідних розділах застосунку.

3.3.2 Управління станом застосунку

Управління станом в мобільних застосунках є важливим аспектом розробки, особливо коли маємо справу зі складними інтерфейсами та великою кількістю даних. Для ефективного управління станом в застосунку було обрано комбінацію Cubit та Provider, яка дозволяє забезпечити чистоту коду, швидку реакцію на зміни та простоту у розширенні функціоналу.

Cubit є ключовим елементом нашої архітектури для управління бізнес-логікою та станом застосунку. Cubit використовується для визначення різних станів, до яких може переходити застосунок, а також для управління переходами між цими станами. Кожен Cubit відповідає за конкретну частину функціоналу застосунку, що дозволяє нам добре розділити його на модулі та зберігати код чистим і структурованим. У папці cubits буде зберігатись управління стану застосунком через Cubit. До прикладу, у ній будуть міститись такі підпапки як: landmark_recognation_cubit, auth_cubit, trip_cubit та інші. У підпапці trip_cubit в свою чергу буде два файли: trip_cubit.dart та trip_state.dart.

Provider забезпечує механізм постачання даних (зазвичай моделей або об'єктів стану) відносно віджетів, які використовують ці дані. Він допомагає передавати дані з Cubit (або будь-якого іншого об'єкта) від батьківського віджета до дочірніх, що дозволяє нам оновлювати і відображати дані у реальному часі без необхідності вручну оновлювати віджети [7]. Буде також створено папка providers, де будуть зберігатись усі стани застосунку, які використовують Provider. У цій папці до прикладу будуть такий файл як dark_theme_provider.dart для зміни теми застосунку.

Ця комбінація Cubit і Provider дозволяє нам побудувати добре структурований та ефективно працюючий застосунок. Ми можемо легко керувати складним станом застосунку, забезпечуючи при цьому відмінну швидкість роботи

та плавну взаємодію з користувачем. Завдяки цьому, застосунок стає більш масштабованим та легко розширюваним, що дозволяє нам швидко впроваджувати новий функціонал і вдосконалювати його відповідно до потреб користувачів.

3.3.3 Кешування даних

Для підвищення продуктивності мобільного застосунку, а також зменшення кількості запитів до серверної частини було застосовано кешування даних у мобільній частині системи. Кешування даних дозволяє нам зменшити час завантаження зображень та зменшити навантаження, що позитивно впливає на швидкість та продуктивність застосунку. У застосунку використовуватимуться пакет octo_image для завантаження та відображення зображень, забезпечуючи при цьому ефективне управління кешуванням даних.

Octo_image автоматично кешує завантажені зображення, що дозволяє нам зберігати їх локально на пристрої користувача для подальшого використання без повторного завантаження. Також використовується пакет shared_preferences для збереження простих даних у вигляді пар «ключ-значення» на пристрої користувача.

Пакет shared_preferences використовуватимуться для кешування певних налаштувань та виборів користувача, таких як мовні установки, тема, останні вибрані місця та інше. Це дозволяє нам зберігати ці дані локально на пристрої користувача та забезпечує швидкий доступ до них без необхідності кожного разу завантажувати їх з мережі.

Використання кешування даних з octo_image та shared_preferences допомагає нам забезпечити швидкий та ефективний доступ до інформації та зображень для наших користувачів, покращуючи їх досвід використання застосунку.

4.4 Приклади найцікавіших алгоритмів та методів

4.4.1 Алгоритм визначення місць

Один з найцікавіших алгоритмів є реалізаціє визначення місць за допомогою технології розпізнавання зображень. Застосунок використовує пакети Flutter для роботи з зображенням та Google Cloud Vision API для розпізнавання визначних

місць на фотографіях. У класі LandmarkRecognition реалізуємо екран, на якому користувач може відкрити камеру або вибрати зображення з галереї для подальшого розпізнавання визначних місць. Клас LandmarkCubit відповідає за логіку обробки зображень та взаємодії з API. При натисканні на кнопку «Open Camera» або «Choose Image from Gallery» викликається метод getImageAndRecognize класу LandmarkCubit. Цей метод відправляє зображення на сервер Google Cloud Vision API для розпізнавання. Після отримання відповіді від сервера ми перевіряємо наявність розпізнаних визначних місць і відображаємо результат на екрані користувача. Цей алгоритм дозволяє користувачам легко визначати визначні місця на своїх фотографіях і отримувати інформацію про них у реальному часі. Використання технології розпізнавання зображень в поєднанні з механізмами Flutter дозволяє нам створити потужний інструмент для дослідження та вивчення світу.

3.4.2 Алгоритм зміни теми застосунку

Також одним з цікавих алгоритмів є реалізація зміни теми застосунку. Щоб реалізувати цю функцію використовується клас DarkThemeProvider з бібліотекою ChangeNotifier. Цей клас відповідає за визначення темної теми в застосунку. Метод darkTheme дозволяє встановити поточну тему на основі значення bool, яке вказує, чи вибрана темна тема. При встановленні нового значення теми відбувається оновлення налаштувань теми за допомогою об'єкта darkThemePreference, після чого сповіщаються всі слухачі про зміни за допомогою методу notifyListeners(). Цей алгоритм дозволяє користувачам зручно переключатись між темним та світлим режимами в застосунку, забезпечуючи при цьому зручний та ефективний інтерфейс.

3.5 Створення UI/UX дизайну системи

У створені зручного та інтуїтивно-зрозумілого інтерфейсу у мобільному застосунку «Tripster» є важливим етапом створити дизайн UI/UX. Процес розробки

складається з декількох етапів для забезпечення та ефективного створення візуальної частини застосунку.

3.5.1 Дослідження та аналіз

Щоб створити зручний та привабливий інтерфейс мобільного застосунку «Tripster» було проведено ретельне дослідження та аналіз існуючих рішень, що мають схожу тематику. Для цього досліджувались популярні мобільні застосунки, такі як TravelSpend, Expedia та TripIt, з метою виявлення найкращих практик, ключових функцій та елементів інтерфейсу, які забезпечують зручність користування. Особлива увага була приділена навігації, розміщенню основних функцій, візуальному вигляду та загальному дизайну інтерфейсу. Цей аналіз допоміг зрозуміти, які елементи є знайомими та інтуїтивно зрозумілими для користувачів, а також виявити можливості для вдосконалення та унікальності.

Також було проаналізовано останні тенденції у сфері дизайну інтерфейсів для користувачів, що дало зрозуміти основні принципи зручного та сучасного дизайну. Було розглянуто найкращі практики створення інтуїтивних та зручних інтерфейсів, принципи організації контенту та навігації, використання кольору, типографіки та іконок. Рекомендації та принципи для розробки інтерфейсу включають:

- забезпечення зручної та інтуїтивної навігації;
- забезпечення адаптивності;
- зосередження на основних функціях та уникнення переобтяження інтерфейсу зайвими елементами;
- зручність введення даних;
- персоналізація та налаштування інтерфейсу.

Ці принципи стали основою для проетування привабливого, зручного та функціонального користувацького інтерфейсу мобільного застосунку «Tripster», що відповідає сучасним вимогам та потребам користувачів у плануванні подорожей.

3.5.2 Інформаційна архітектура та прототипування

При проектуванні мобільного застосунку «Tripster» приділялося особливу увагу створенню зручної та логічної інформаційної архітектури, а також прототипуванню для визначення структури та функціональності застосунку.

Інформаційна архітектура застосунку «Tripster» включає в себе структуру даних та способи їх організації для забезпечення легкості навігації та зручності користування. Було ретельно проаналізовано всі функції та елементи, які має містити застосунок, і розмістили їх у логічну ієрархію, що відображає потреби користувача та послідовність їх використання. На вершині ієрархії знаходиться головний екран, який вітає користувача та надає доступ до основних функцій застосунку, таких як планування подорожей, перегляд рекомендованих місць та інші. З цього екрану користувач може легко переходити до розділів, що відповідають його поточним потребам. Далі, функціональність застосунку поділена на кілька ключових розділів, таких як «Мої подорожі», «Мій профіль», «Рекомендовані місця» та «Камера». Кожен з цих розділів має компоненти, де користувач може здійснювати конкретні дії, такі як перегляд деталей подорожі, додавання нових місць до списку вподобань, створення нових планів подорожей, збереження нотаток та ін.

Крім того, важливою складовою ієрархії є можливість переходу між розділами та функціями застосунку за допомогою інтуїтивних навігаційних елементів, таких як меню, вкладки або кнопки «назад». Це дозволяє користувачеві легко переміщатися в застосунку та зручно використовувати всі його можливості. Така організація дозволяє забезпечити логічну та структуровану ієрархію, яка спрощує взаємодію з застосунком і забезпечує зручність користування для кожного користувача.

Перед розробкою запропонованої фінальної версії застосунку створюються прототипи, які відображають вигляд та функціонал кожного екрану. Це дозволяє перевірити взаємодію елементів інтерфейсу, розміщення контенту та загальну зручність використання застосунку. На рисунку 4.1 зображено прототип авторизації, профілю користувача та створеного альбому.

Рисунок 3.1 – Прототип авторизації, профілю користувача та альбому фотографій (Виконана самостійно)

Прототип авторизації відображає два текстових поля для введення інформації користувача, а також кнопку для входу у застосунок. Також у користувача є можливість перейти у екран реєстрації, якщо він не зареєстрований у системі. Прототип профілю відображає сторінку користувача, де зберігається його основна інформація, а саме: ім’я та пошта. Також у користувача є змога редагувати профіль та перейти у налаштування. Одним з головних функцій – є відображення створених альбомів та додати альбом. Коли користувач переходить на створений альбом, у нього є можливість додати фото до альбому та переглянути їх. Завдяки якісній інформаційній архітектурі та добре розроблених прототипів створюється основа для успішної реалізації проекту, а також забезпечується позитивний досвід у користувачів.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис структури проекту

Після аналізу предметної області та проектування було реалізовано застосунок «Tripster», що здатен допомогти користувачу у процесі планування та організації подорожей. Для створення застосунку використовувався фреймворк Flutter, який забезпечує не лише високу швидкість та ефективність, але й дозволяє реалізувати привабливий та сучасний дизайн. Розробка коду проводилася в інтегрованому середовищі розробки – Visual Studio Code, яке надає розширені можливості для аналізу та оптимізації коду, що допомагає забезпечити якість та надійність застосунку. У додатку А відображено дерево, що містить у собі 32 директорії та 75 файлів.

Реалізований проект складається з трьох папок: data, domain та presentation, кожна з яких відповідає за певні аспекти функціональності та архітектури застосунку.

Папка «data» містить репозиторії, які відповідають за взаємодію з даними додатку. Кожен репозиторій відповідає за конкретну сутність додатку, таку як аутентифікація, галерея, пам'ятки тощо.

У папці «domain» розміщуються моделі даних, які використовуються в застосунку. Ці моделі представляють основні об'єкти та сутності, з якими працює застосунок, такі як фотографії галереї, користувачі, місця тощо.

Папка «presentation» містить весь вихідний код, який відображається на інтерфейсі користувача. Це включає в себе всі віджети, екрани, управління станом, а також інші компоненти, необхідні для представлення даних та взаємодії з користувачем.

Крім того, у папці «utils» знаходяться додаткові утиліти та допоміжні файли, такі як константи, налаштування теми, робота з локалізацією тощо.

Ця структура допомагає зробити проект більш масштабованим і підтримуваним. Розділення коду на логічні модулі полегшує роботу над окремими частинами додатку, що сприяє швидкому виявленню та виправленню помилок. Крім того, вона сприяє збереженню стабільності додатку під час внесення змін,

оскільки розробники можуть зосередитися на конкретних аспектах без впливу на інші частини коду. Такий підхід також полегшує спільну роботу декількох розробників над проектом та сприяє його подальшому розвитку з часом.

4.2 Використані бібліотеки

Програмний код застосунку «Tripster», був доповнений рядом бібліотек та плагінів, що значно розширили його функціональні можливості та покращили загальний користувацький досвід. Використання цих інструментів дозволило швидше та ефективніше реалізувати ключові функції застосунку, забезпечити високу продуктивність, зручний інтерфейс та взаємодію з користувачами.

Нижче наведено опис основних бібліотек та плагінів, використаних у проекті:

1. flutter_bloc – бібліотека для управління станом застосунку з використанням патерну BLoC (Business Logic Component). Вона забезпечує чисту архітектуру, розділяючи бізнес-логіку від UI, що сприяє легкому тестуванню та підтримці коду;
2. google_maps_flutter – плагін для інтеграції карт Google Maps у застосунок. Це дозволяє відображати карти та додавати маркери, що є важливими для планування подорожей;
3. easy_localization – плагін для реалізації багатомовності у додатку. Він дозволяє легко додавати підтримку різних мов, що робить застосунок доступним для користувачів з різних країн;
4. octo_image – бібліотека для відображення зображень, які завантажуються з мережі та зберегаються в кеші;
5. geolocator – плагін для отримання геолокаційних даних. Він дозволяє визначати поточне місцезнаходження користувача, що є критичним для застосунку, пов'язаних з подорожами та навігацією;
6. sharedPreferences – це плагін для зберігання простих даних у формі пар ключ-значення. Він дозволяє зручно зберігати та отримувати дані;

7. google Vision API – сервіс, який забезпечує потужні можливості для аналізу зображень. Зокрема, у застосунку «Tripster» використовується функція виявлення орієнтирів (landmark detection), яка дозволяє автоматично розпізнавати відомі місця на фотографіях.

4.3 Авторизація у програмному застосунку

Для того, щоб користувач міг отримати доступ до функцій застосунку, необхідно пройти процес авторизації. Авторизація забезпечує безпеку особистих даних користувачів. Користувачі можуть здійснити реєстрацію через свою пошту. Зареєструвавшись, він має змогу також увійти під цими даними на сайті Tripster. Незареєстровані користувачі можуть увійти в систему, використовуючи свій логін та пароль.

Після успішної авторизації користувачу видається токен, який використовується для подальшої автентифікації в системі. Токен зберігається та додається до кожного запиту до сервера для підтвердження автентичності користувача.

Всі введені дані перевіряються на коректність. Зокрема, перевіряються формат електронної пошти, унікальність облікового запису, а також чи заповнені усі обов'язкові поля. Приклад коду валідації на формат електронної пошти:

```
TextFormField(
validator: (value) {
   if (value!.isEmpty) {
       return LocaleKeys.invalid_email.tr();
        } else if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$')
              .hasMatch(value)) {
                return LocaleKeys.invalid_email.tr();
                  }
                      return null;
                    },
```

Умова if (value!.isEmpty) перевіряє, чи є значення поля порожнім (isEmpty). Якщо так, то повертається повідомлення про те, що електронна пошта недійсна. Умова else if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$').hasMatch(value)) перевіряє, чи введений текст відповідає правильному формату електронної пошти за допомогою регулярного виразу. Якщо текст не відповідає формату, повертається

повідомлення про недійсну електронну адресу. Якщо жодна з вищезазначених умов не виконується, то повертається null, що означає, що введені дані відповідають усім вимогам, і форма є валідною. На рисунку 4.3 зображено екран реєстрації та екран логіну.

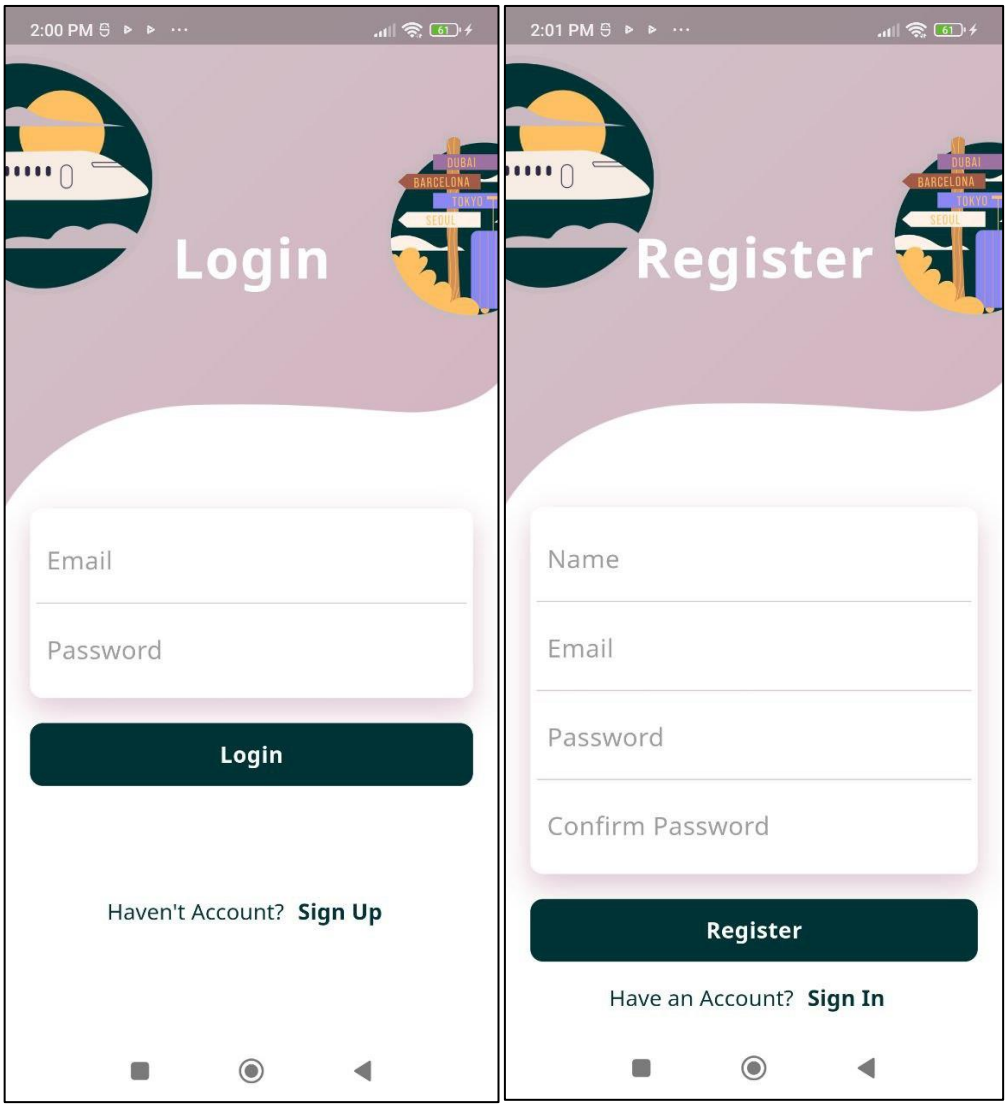


Рисунок 4.1 – Екран логіну та реєстрації

Процес авторизації є критично важливим для забезпечення безпеки особистих даних користувачів у додатку Tripster. Реалізація авторизації через електронну пошту дозволяє користувачам легко реєструватися та входити в систему. Виділення токену після успішної авторизації значно підвищує рівень безпеки, забезпечуючи, що тільки автентифіковані користувачі можуть отримати доступ до функцій додатку.

4.4 Цікаві місця та рекомендації у програмному застосунку

У користувача після успішної авторизації у програмний застосунок є можливість переглянути цікаві місця для відвідування, а також переглянути рекомендації. Ця функціональність забезпечує користувачам доступ до найкращих туристичних місць, що допомагає їм планувати свої подорожі більш ефективно. Для кожного місця надається детальна інформація, включаючи опис, фотографію та точне місцезнаходження. На рисунку 4.2 зображено усі цікаві місця та їх детальна інформація.

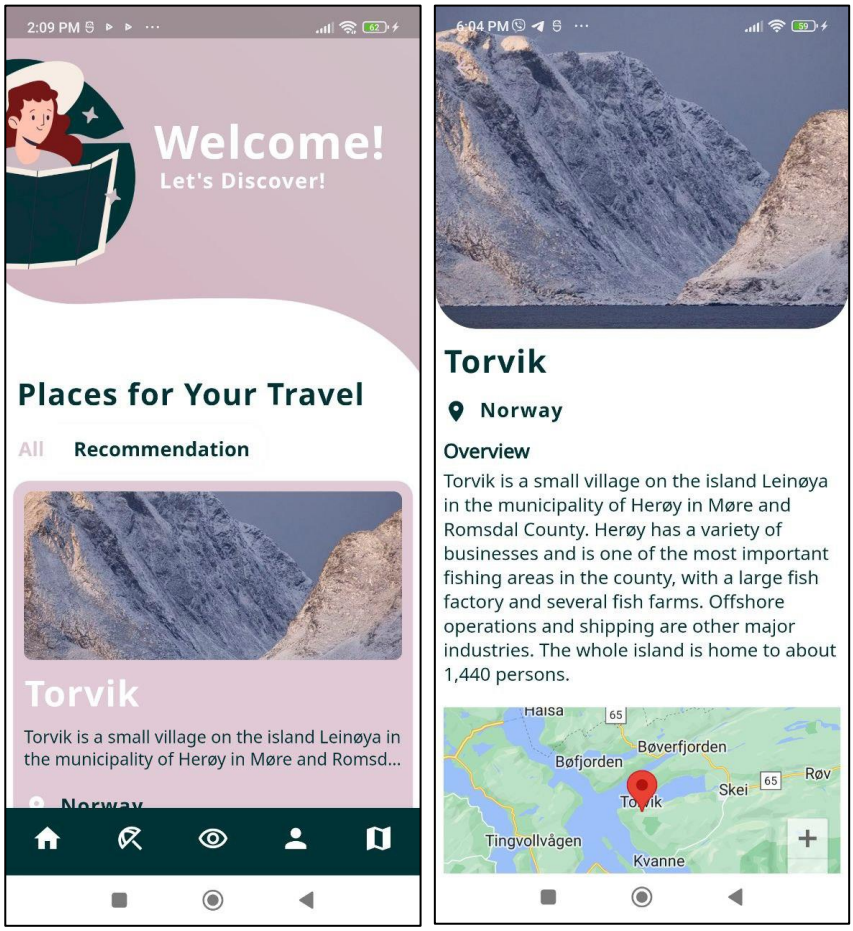


Рисунок 4.2 – Цікаві місця та детальна інформація

Для персоналізованих рекомендацій користувачу потрібно пройти тестування на клієнтській частині програмного застосунку. Це тестування може включати питання щодо вподобань у подорожах, улюблених видів активностей, типу місць, які користувач віддає перевагу (наприклад, музеї, парки, ресторани), та

інші критерії. На основі цих тестувань серверна частина аналізує відповіді та повертає дописи, що найкраще відповідають інтересам користувача.

Для реалізації отримання рекомендованих дописів з серверної частини програмної системи використовувався код нижче:

```
Future<List<Place>> getRecommendation(String? token) async {
          final response = await http.get(
            Uri.parse('https://tripser-
backend.onrender.com/recomendations'),
            headers: {'Authorization': 'Bearer $token'},
          );
          if (response.statusCode == 200) {
            final Map<String, dynamic> jsonData =
json.decode(response.body);
            final List<dynamic> placeGroups = jsonData['placeGroups'];
            List<Place> allPlaces = [];
            for (var group in placeGroups) {
              final List<dynamic> placesData =
group['placesCollection']['items'];
              List<Place> places =
                  placesData.map((json) => Place.fromJson(json)).toList();
              allPlaces.addAll(places);
            }
            return allPlaces;
          } else {
            throw Exception('Failed to load places:
${response.statusCode}');
          }
        }
```

Реалізований метод асинхронно отримує рекомендації місць для відвідування з сервера. Після успішної авторизації користувача у програмний застосунок, вона виконує HTTP GET-запит до вказаного URL з переданим токеном авторизації у заголовках. Після отримання відповіді в форматі JSON функція обробляє дані, виділяючи з них список груп місць, які включають колекції рекомендованих місць. Далі дані кожного місця перетворюються у відповідні об'єкти Dart за допомогою методу Place.fromJson. Усі отримані місця додаються до загального списку, який повертається як результат функції. У випадку невдалого запиту або отримання невідповідного статус-коду, генерується виняток з відповідним повідомленням про помилку. Метод забезпечує безпроблемне отримання персоналізованих рекомендацій місць для відвідування користувачем у програмі.

4.5 Розпізнавання місць у програмному застосунку

Для кращої взаємодії з застосунком було реалізовано розпізнавання місць за допомогою Google Vision API. Користувач має змогу завантажити фотографію з пам’яткою та отримати повну назву цієї пам’ятки. На рисунку 4.3 відображено вигляд екранів розпізнавання місць.

Рисунок 4.3 – Екрани розпізнавання місць у програмному застосунку

Після завантаження фотографії система використовує Google Vision API для аналізу зображення та визначення об'єктів на ньому. Після розпізнавання система повертає користувачеві повну назву пам'ятки або об'єкта, який відображений на фотографії. Це дозволяє швидко і точно ідентифікувати цікаві місця навколо, навіть

якщо користувач не знайомий з ними. Реалізація отримання місця за допомогою API знаходиться у додатку Б.

Спочатку функція отримує зображення у форматі XFile та зчитує його в байтовий масив. Після цього виконується POST-запит до API Google Vision, де вказується тип аналізу як LANDMARK_DETECTION та передається зображення у кодованому форматі base64. У випадку успішного виконання запиту з кодом статусу 200, функція отримує відповідь у форматі JSON та аналізує її. Якщо пам'ятка була розпізнана, дані про неї зберігаються у відповідному об'єкті Landmark, який містить опис та зображення пам'ятки. У випадку, якщо пам'ятка не була розпізнана або відсутня у відповіді, повертається об'єкт Landmark із описом «Landmark not found». В разі невдалого запиту, функція генерує виняток з відповідним повідомленням про помилку та кодом статусу.

Ця функція дозволяє користувачам швидко та зручно отримувати інформацію про пам'ятки на фотографіях, що покращує їхнє взаємодію з застосунком та робить процес відкриття нових місць більш захоплюючим та цікавим.

4.6 Подорожі у програмному застосунку

Однією з ключових функціональних можливостей додатку «Tripster» є його здатність допомагати користувачам ефективно планувати та керувати своїми подорожами. Користувачі можуть переглядати створені ними подорожі на клієнтській частині програмної системи, отримуючи доступ до детальної інформації про кожну з них. Основні аспекти, такі як бюджет подорожі, дата початку та закінчення, а також місце призначення, відображаються для кожної подорожі. Місце призначення відображається за допомогою Google Maps та відображається для двох екранів: списку подорожей та детальної інформації.

Також користувачі можуть здійснювати пошук по країні у своїх подорожах, а також сортувати їх по даті.

У межах кожної подорожі користувачі можуть переглянути окремі дні, розподіляючи подорож на відрізки часу та плануючи активності на кожен з них.

Дні включають важливу інформацію, таку як бюджет на день, дата, список місць, які користувач планує відвідати, а також детальний план дій на кожен конкретний день. Це дозволяє користувачам більш точно та ефективно планувати свої подорожі, розподіляючи час та ресурси відповідно до їхніх потреб та побажань. Користувач може взаємодіяти з планом дня, а саме редагувати, оновлювати та видаляти дані з плану. На рисунку 4.4 відображено список подорожей, його детальна інформація та список днів у подорожі.

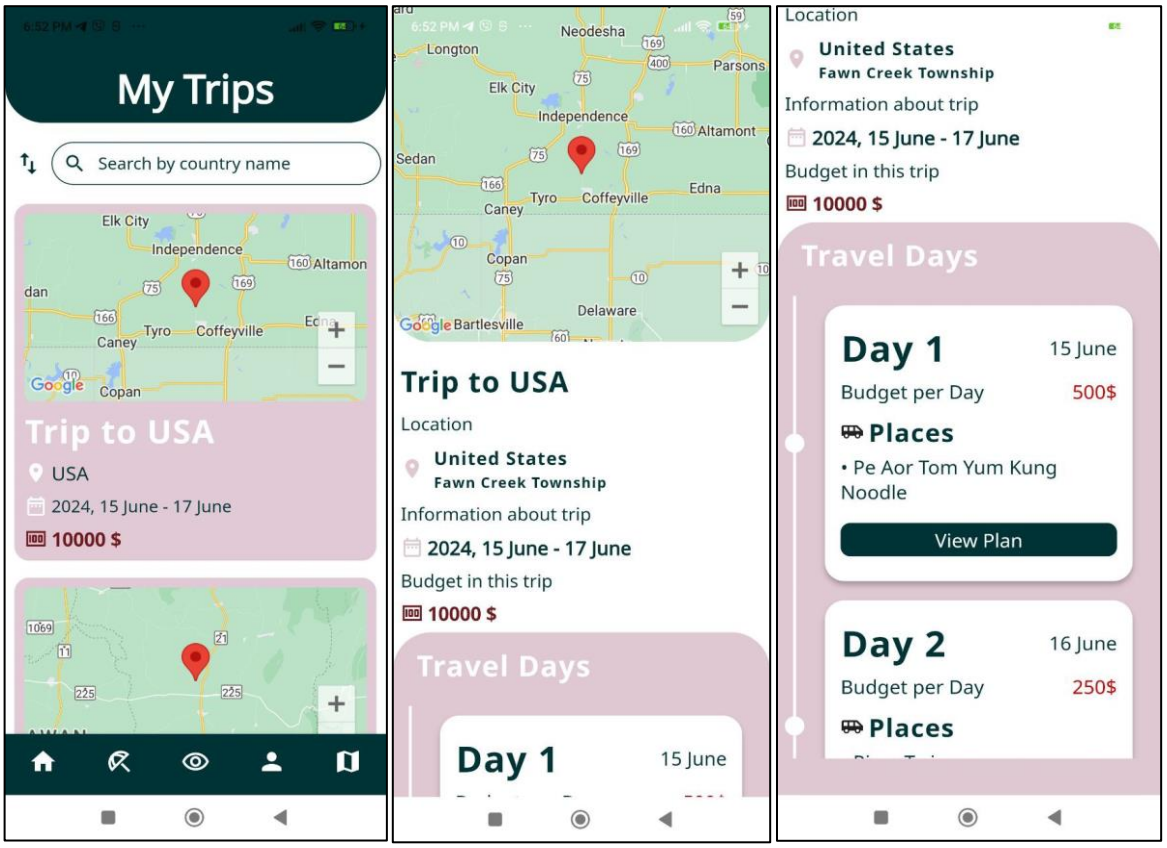


Рисунок 4.4 – Список подорожей та днів у конкретній подорожі

Крім того, функціонал дозволяє користувачам взаємодіяти з планом кожного дня, надаючи їм можливість редагувати, оновлювати та видаляти дані, щоб відповідати змінам планів або уточненим уявленням про подорож. Ця гнучкість дозволяє користувачам легко адаптувати свої плани до непередбачених обставин або нових можливостей, забезпечуючи їм найкращий досвід подорожей.

Відображення планування конкретного дня у подорожі можна знайти у зображенні, представленому на рисунку 4.5.

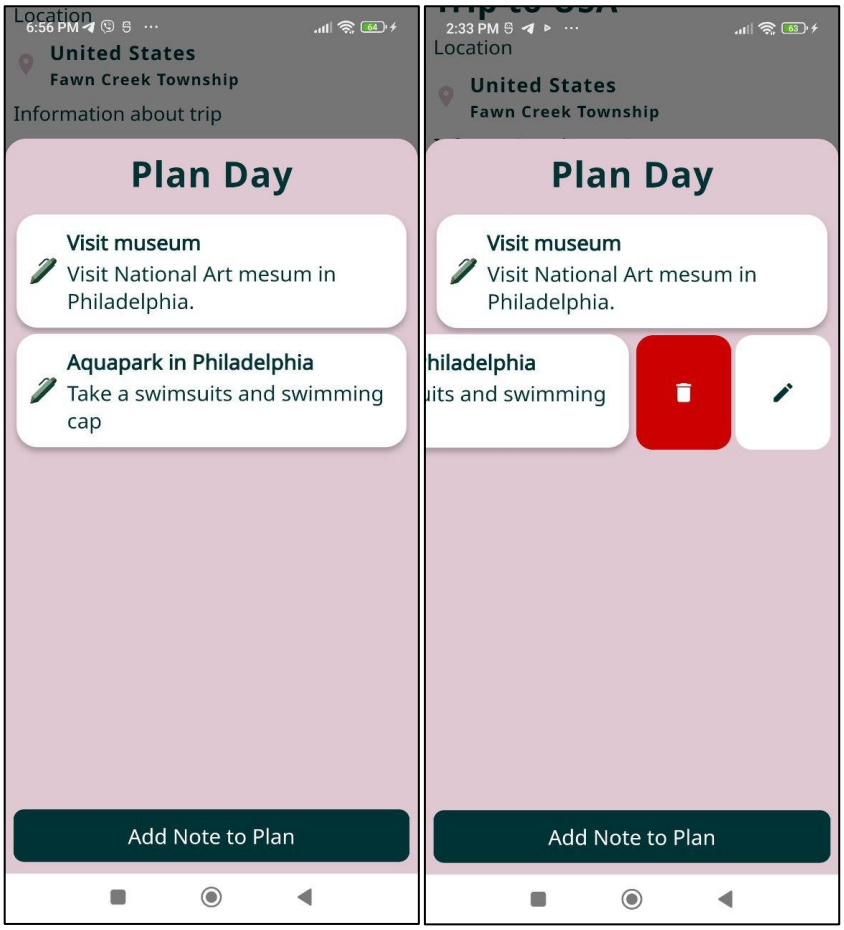


Рисунок 4.5 – Планування конкретного дня

Цей фрагмент коду відповідає за створення нового допису до планування конкретного дня у подорожі. Його реалізація виглядає наступним чином:

```
Future<Note> createNote({
    required String title,
    required String description,
    required String vacationDayId,
    required String? token,
  }) async {
final response = await http.post(
      Uri.parse('https://tripser-backend.onrender.com/notes/'),
      headers: {
        'Authorization': 'Bearer $token',
        'Content-Type': 'application/json',
      },
body: json.encode({
        'title': title,
        'description': description,
        'vacationDayId': vacationDayId,
      }),
    );

    if (response.statusCode == 201) {
      final jsonData = json.decode(response.body);
      return Note.fromJson(jsonData);
```

```
        } else {
          throw Exception('Failed to create note:
${response.statusCode}');
        }
```

Ця функція createNote відповідає за створення нового допису у відповідності до наданих параметрів. Вона виконує POST-запит до вказаної URL з вказаними заголовками та тілом запиту, яке містить дані про замітку у форматі JSON. Після успішного створення запису, вона повертає об'єкт типу Note, який представляє новостворений допис. У випадку помилки, вона генерує виняток з відповідним повідомленням про невдале створення замітки та кодом статусу відповіді.

Таким чином, функціонал подорожей в застосунку «Tripster» забезпечує користувачам організовану подорож та допомагає в плануванні. Завдяки цьому функціоналу користувачі можуть легко маніпулювати своїми подорожами, розподіляти їх на дні та керувати різними аспектами своїх подорожей. Вони можуть змінювати та оновлювати свої плани в разі необхідності. Цей функціонал робить процес організації подорожей більш простим, зручним та ефективним для користувачів, допомагаючи їм насолоджуватися кожним моментом своєї подорожі.

4.7 Профіль користувача та кастомізація застосунку

Після успішної реєстрації у застосунку користувач автоматично отримує персоналізований профіль, який містить важливі особисті дані. Цей профіль створюється з метою забезпечення зручності та персоналізації досвіду користувача. Він містить фотографію за замовчуванням, яку користувач може легко змінити на власну, а також включає ім'я та електронну адресу, які були вказані під час реєстрації.

Одним із ключових елементів цього профілю є можливість редагувати особисту інформацію. Користувачеві надається можливість змінити свою фотографію та ім'я, щоб вони краще відображали його особистість та індивідуальність. Це дозволяє користувачам персоналізувати свій профіль та відчувати себе більш комфортно в межах застосунку. Щоб редагувати свій профіль,

користувачу потрібно натиснути на «Edit Profile» або іконку редагування. Профіль, а також форма редагування зображена на рисунку 4.6.

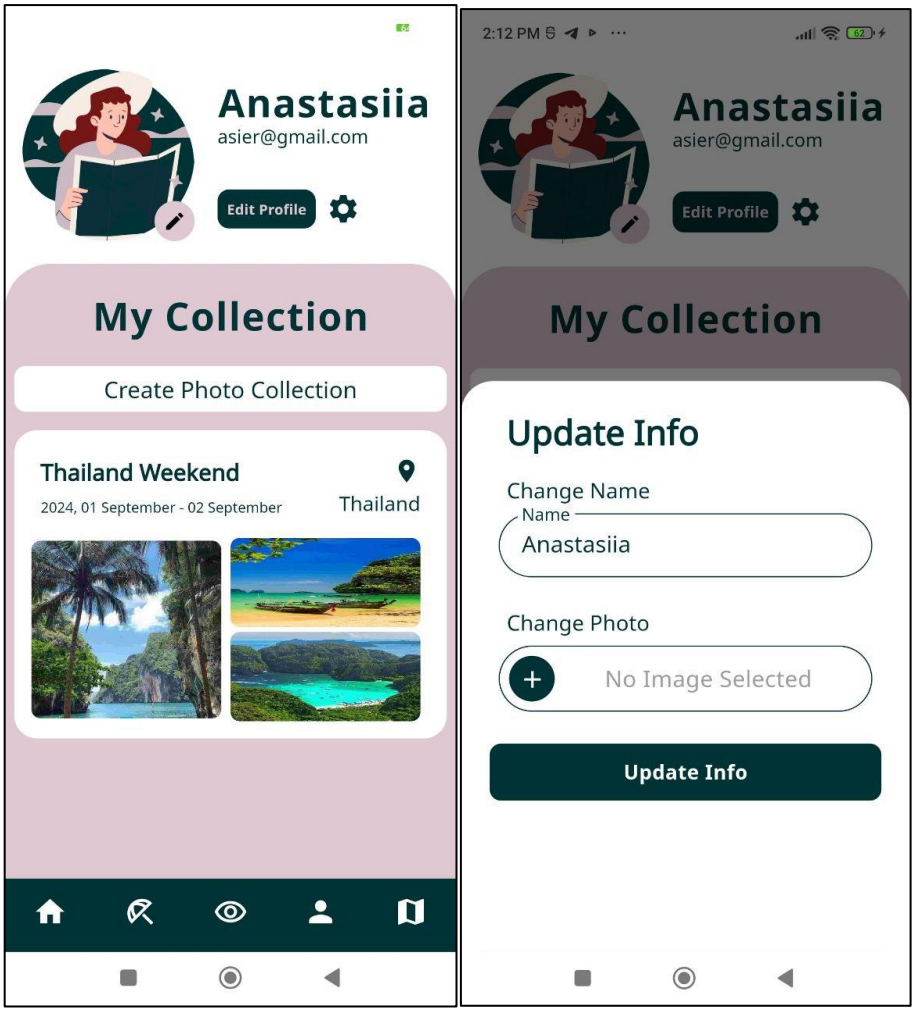


Рисунок 4.6 – Профіль та форма редагування

Також у користувача є можливість кастомізувати застосунок під власні потреби. Можливість кастомізації застосунку під власні потреби є важливим аспектом для забезпечення комфортного та особистого досвіду користувача. За допомогою цієї функціональності користувачі мають змогу адаптувати застосунок до своїх власних уподобань та потреб.

Один з важливих елементів кастомізації – це можливість користувачів обирати тему застосунку, включаючи світлу та темну, відповідно до їхніх вподобань та стилю. Наприклад, деякі користувачі можуть віддавати перевагу темному режиму для збереження зору, тоді як інші виберуть світлий режим для

більшого комфорту читання. Зміна кольору теми знаходиться у сторінці «Налаштування» та має наступний вигляд (Рис. 4.7):

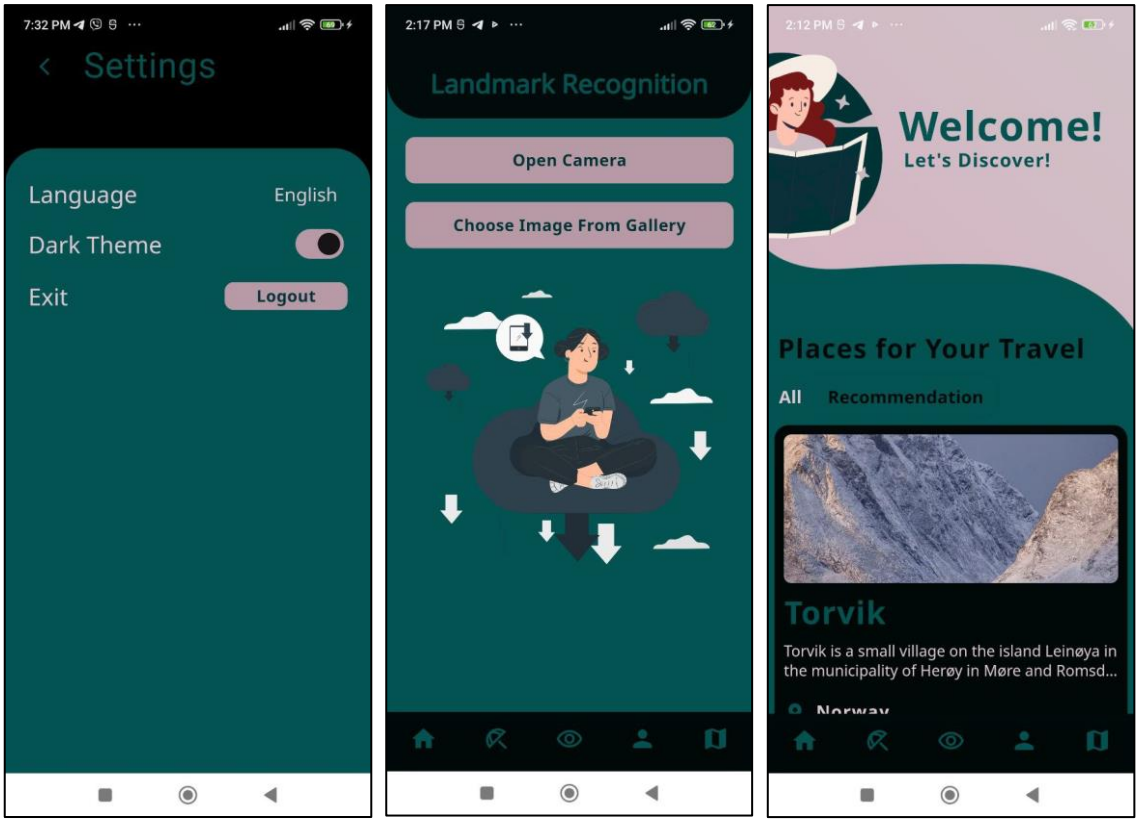


Рисунок 4.7 – Зміна теми та вигляд зміненої теми

Додатково, користувачі мають можливість обирати мову застосунку з двох доступних варіантів: української та англійської. Ця функціональність особливо важлива для аудиторії з різних країн та мовних спільнот, оскільки дозволяє забезпечити доступність додатку для широкого кола користувачів з різних культурних та мовних середовищ. Також використання мови, яка більш зрозуміла або комфортна для користувача, сприяє покращенню його взаємодії з застосунком та забезпечує більш ефективне використання всіма його можливостями. Такий підхід допомагає створити дружнє та приємне середовище для всіх користувачів, незалежно від їхнього рівня знання мови або мовних уподобань. Це зображено на рисунку 4.8.

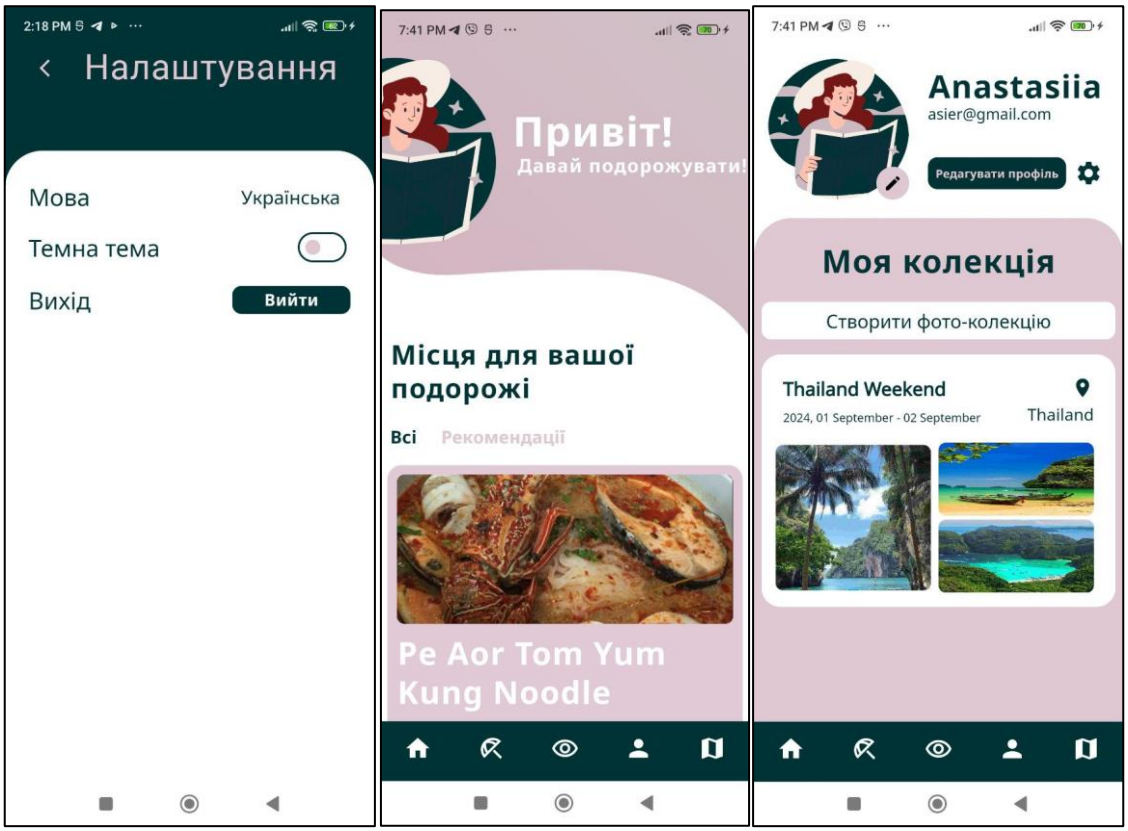


Рисунок 4.8 – Зміна мови та вигляд зміненої мови

Для забезпечення локалізації у застосунку було використано потужний та зручний пакет EasyLocalization. Цей пакет надає простий та ефективний спосіб реалізації перекладу текстових ресурсів у різні мови. Після підключення пакету до проекту, було почато процес локалізації, що дозволило застосунку стати доступним для користувачів з різних країн та мовних середовищ. У головному файлі main.dart було виконано налаштування, необхідні для правильної роботи локалізації:

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await EasyLocalization.ensureInitialized();
  tz.initializeTimeZones();
  runApp(
    EasyLocalization(
        supportedLocales: const [
          Locale('en'),
          Locale('uk'),
        ],
        path: 'assets/translations',
        fallbackLocale: const Locale('en'),
        assetLoader: const CodegenLoader(),
        child: MyApp(token: token)),
  )
}
```

Це включало ініціалізацію пакету та налаштування основних параметрів, таких як мова за замовчуванням, доступні мови, шлях до ресурсів перекладу тощо. Після цього можна легко додавати текстові ресурси до файлів з перекладами для кожної мови, що дозволяло додатку автоматично вибирати та відображати відповідні тексти в залежності від вибраної користувачем мови.

На завершення, важливо зазначити, що можливість кастомізації мови зміни теми допомагає створити застосунок який відповідає потребам і вимогам різноманітної аудиторії. Це робить досвід користування застосунком більш приємним, комфортним та доступним для всіх користувачів. Будь-яка можливість, яка дозволяє користувачам персоналізувати застосунок під свої індивідуальні потреби, сприяє збільшенню задоволення від використання та підвищує загальний рівень задоволення від користування.

4.8 Фото-колекції у програмному застосунку

Функція створення фото-колекцій є однією з ключових можливостей розробленого застосунку. Вона дозволяє користувачам збирати свої спогади та враження у вигляді фотографій, пов'язаних з конкретними подорожами. Користувачі мають можливість створити нову колекцію, вибравши одну зі своїх подорожей як основу, а потім додавати до неї фотографії з різних моментів своєї подорожі. Також є можливість редагування колекції (додавши нові фотографії), та також видалення самої колекції.

Кожна фото-колекція створюється з урахуванням конкретної подорожі, що дозволяє користувачам легко організовувати та зберігати фотографії відповідно до контексту. Після створення колекції користувач може додавати нові фотографії, вибиравши їх з галереї, натиснувши на іконку галереї, а також видаляти прямо з колекції, затиснувши фотографію.

На рисунку 4.9 зображено: на першому екрані відображення додавання нової галереї, де користувач обирає подорож зі списку власних, а також додає до них фотографії, на другому екрані показано відображення створеної галереї, а на третьому екрані показано детальний екран самої фото-колекції.

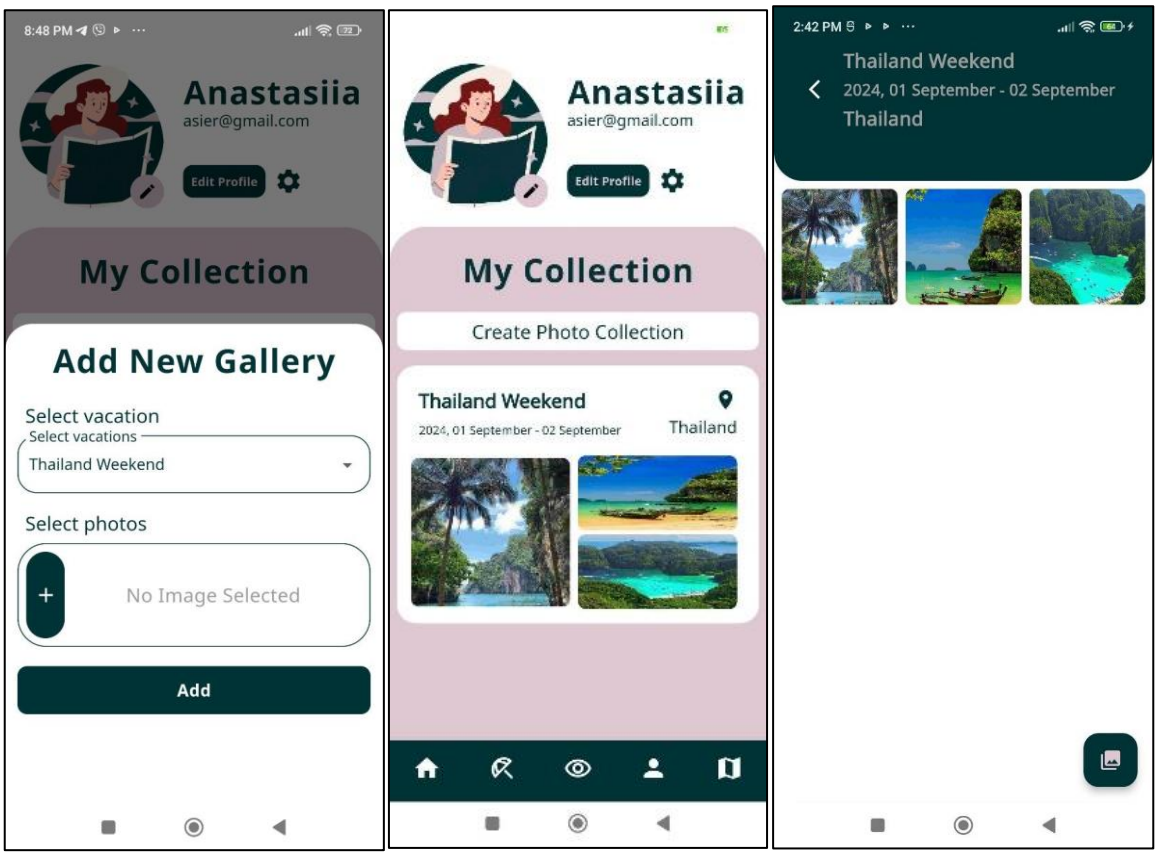


Рисунок 4.9 – Створення галереї та її вигляд

Для передачі створеної фото-колекції разом з фотографіями на сервер був розроблений метод createGallery. Цей метод використовується для інтеграції з серверною частиною застосунку та дозволяє користувачам зберігати свої фото-колекції. Він приймає список файлів зображень, ідентифікатор подорожі та токен для автентифікації, а потім відправляє ці дані на сервер за допомогою POST-запиту. Написаний метод createGallery, що наведений нижче:

```
Future<Gallery> createGallery({
     required List<File> images,
     required String vacationId,
     required String? token,
     }) async {
     try {
     var request = http.MultipartRequest(
     'POST',
     Uri.parse('https://tripser-
backend.onrender.com/vacations/gallery'),);
     request.headers.addAll({
     'Authorization': 'Bearer $token',});
     request.fields['vacationId'] = vacationId;
     for (var image in images) {
     request.files
```

```
      .add(await http.MultipartFile.fromPath('image', image.path));}
      var response = await http.Response.fromStream(await
request.send());
      final jsonData = json.decode(response.body);
      return Gallery.fromJson(jsonData);
      } catch (e) {
      throw Exception('Failed to create gallery: ${e}');}
      }
```

Функція createGallery використовує об'єкт MultipartRequest з пакету http для створення складного запиту типу POST, який може передавати файли разом із звичайними полями форми. Вона формує запит, додаючи до нього дані про подорож та зображення. Потім запит відправляється на сервер, де дані обробляються. Використання MultipartRequest дозволяє ефективно передавати багато зображень разом з іншими даними, забезпечуючи потрібну функціональність для створення фото-колекції у застосунку.

Фото-колекції у застосунку додають значну цінність для користувачів, допомагаючи їм зберігати та організовувати свої враження від подорожей. Колекції дають змогу створювати запам'ятовуючі фото-історії, які залишають невимірні враження від подорожей. Крім того, функція фото-колекцій робить застосунок більш інтерактивним і особистим.

Крім того, функціонал фото-колекцій не лише дозволяє користувачам організувати свої фотографії за подорожами, але й надає можливість керувати ними. Користувачі можуть видаляти, оновлювати та додавати нові фотографії до колекцій, надаючи їм повний контроль над своїми враженнями від подорожей. Це дозволяє створювати актуальні та завершені історії своїх подорожей, що є важливим аспектом для подорожуючих, які прагнуть зберігати та ділитися своїми спогадами.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування програмного застосунку

Для тестування розробленого застосунку «Tripster» використовувалося мануальне тестування. Мануальне тестування застосунку зосереджувалося на різноманітних аспектах його функціонування, щоб забезпечити максимальну якість і стабільність роботи. Зокрема, перевірка функціональності включала тестування основних операцій, таких як авторизація, реєстрація та профілем корситувача. Це передбачало ретельне тестування процесів входу в систему та створення нових акаунтів, щоб гарантувати, що користувачі можуть безперешкодно зареєструватися та увійти до свого облікового запису.

Крім цього, була проведена детальна перевірка функцій редагування профілю, зокрема можливостей зміни профільної фотографії та імені користувача. Тестування включало завантаження нової фотографії, перевірку підтримуваних форматів файлів (JPEG, PNG тощо) та максимального розміру файлу, а також перевірку правильності відображення нової фотографії після збереження. Окрім цього, було протестовано можливість зміни імені користувача, перевірено валідність введених даних та коректність збереження нової інформації.

У таблиці 5.1 відображено тест-кейс з реєстрацією нового користувача у застосунку «Tripster».

Таблиця 5.1 – Тест-кейс №1 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Реєстрація нового користувача		
Власник тесту:	Вельма Анастасія Олександрівна		
Дата створення:	01.06.2024		
Мета тесту:	Перевірка процесу реєстрації нового користувача у застосунку		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити завантажений застосунок	Користувач має доступ до застосунку	Пройдено

Кінець таблиці 5.1

Реєстрація нового користувача			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити застосунок «Tripster»	Застосунок відкривається і відображається головний екран з кнопкою «Реєстрація».	Пройдено
2	Натиснути на кнопку «Реєстрація».	Відкривається форма реєстрації з полями для введення даних.	Пройдено
3	Заповнити обов’язкове поле «Ім’я»	Поле заповнене, кількість символів правильна, значення збережене.	Пройдено
4	Заповнити обов’язкове поле «Пошта»	Поле заповнене та пройшло валідацію на коректність (правильний формат пошти).	Пройдено
4	Заповнити обов’язкове поле «Пароль»	Поле заповнене, пароль відповідає вимогам безпеки (не менше 8 символів, містить літери та цифри)	Пройдено
5	Заповнити обов’язкове поле «Підтвердити пароль».	Поле заповнене, введений пароль збігається з паролем у попередньому полі.	Пройдено
6	Натиснути кнопку «Зареєструватися»	Застосунок відправляє дані на сервер для реєстрації нового користувача. Після успішної реєстрації, користувач автоматично входить у систему.	Пройдено
Результати тестування			
Тестувальник: Вельма А.О.	Дата прогону тесту: 01.06.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

У таблиці 5.2 відображено тест-кейс з авторизацією користувача у застосунку «Tripster».

Таблиця 5.2 – Тест-кейс №2 (таблиця виконана самостійно)

Інформація про тест-кейс	
Ідентифікатор тесту:	Тест-кейс №2
Опис функції:	Авторизація користувача
Власник тесту:	Вельма Анастасія Олександрівна
Дата створення:	01.06.2024
Мета тесту:	Перевірка процесу авторизації користувача у застосунку

Кінець таблиці 5.2

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити завантажений застосунок	Користувач має доступ до застосунку	Пройдено
Авторизація користувача			
№	Опис випадку	Очікуваний результат	Висновок
	Відкрити застосунок «Tripster»	Застосунок відкривається і відображається головний екран з кнопкою «Вхід».	Пройдено
1	Натиснути на кнопку «Вхід».	Відкривається форма для авторизації з полями для введення даних.	Пройдено
3	Заповнити обов’язкове поле «Пошта»	Поле заповнене та пройшло валідацію на коректність (правильний формат пошти та чи існування такої пошти у застосунку).	Пройдено
4	Заповнити обов’язкове поле «Пароль»	Поле заповнене та коректно введене	Пройдено
5	Натиснути кнопку «Вхід»	Застосунок отримує дані з серверу для входу користувача. Після успішної авторизації, користувач автоматично входить у систему та відображається меню застосунку.	Пройдено
Результати тестування			
Тестувальник: Вельма А.О.	Дата прогону тесту: 01.06.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

У таблиці 5.3 відображено тест-кейс з редагуванням профілю користувача у застосунку «Tripster».

Таблиця 5.3 – Тест-кейс №3 (таблиця виконана самостійно)

Інформація про тест-кейс	
Ідентифікатор тесту:	Тест-кейс №3
Опис функції:	Редагування профілю користувача
Власник тесту:	Вельма Анастасія Олександрівна
Дата створення:	01.06.2024
Мета тесту:	Перевірка процесу редагування профілю користувача у застосунку

Кінець таблиці 5.3

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Користувач зареєстрований у системі та увійшов у свій обліковий запис	Користувач увійшов у свій обліковий запис та перейшов на сторінку з меню	Пройдено
2	Перейти до сторінки редагування профілю	Користувач знаходиться на сторінці редагування профілю	Пройдено
Редагування профілю користувача			
№	Опис випадку	Очікуваний результат	Висновок
	Ввести нове ім'я користувача у відповідне поле.	Нове ім'я введено у поле і відображається правильно	Пройдено
1	Натиснути на кнопку «Завантажити нову фотографію»	Здійснено перехід до галереї користувача	Пройдено
3	Вибрати файл з новою фотографією	Файл обрано і відображається попередній перегляд нової фотографії. Файл підходить під формат JPEG або PNG	Пройдено
4	Натиснути на кнопку «Оновити»	Застосунок оновлює дані про користувача та зберігає них.	Пройдено
Результати тестування			
Тестувальник: Вельма А.О.	Дата прогону тесту: 01.06.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Отже, можна сказати, що тестування застосунку «Tripster» підтвердило його високу якість і стабільність функціонування. Мануальне тестування дозволило перевірити різноманітні аспекти, включаючи основні операції, такі як авторизація, реєстрація та робота з профілем користувача. Кожен тест-кейс був успішно пройдений, що свідчить про правильну реалізацію функціональності та коректну роботу застосунку в цілому. Всі перевірені процеси, включаючи реєстрацію нових користувачів, авторизацію та редагування профілю, працюють згідно з очікуваннями та вимогами. Такий підхід до тестування гарантує високу якість та задоволення потреб користувачів у використанні застосунку «Tripster».

ВИСНОВКИ

Під час роботи над дипломним проектом «Програмна система для подорожей визначними місцями. Мобільний застосунок» був успішно реалізований мобільний застосунок, що допомагає користувачам якісно та зручно організовувати та планувати свої подорожі. Впродовж даного дипломного проекту була досягнута головна мета – створення застосунку зі зручним та інтуїтивно-зрозумілим користувальницьким інтерфейсом, що спрощує планування подорожі та надає необхідний функціонал для пошуку, збереження та організації про визначні місця. Для досягнення даної мети, було ознайомлено з можливими варіантами архітектур, вивчено інформацію про архітектурні патерни проектування, а також було проаналізовано предметну область та розроблено проектну специфікацію, зроблено проектування та прототипування, а також реалізовано застосунок, використовуючи принципи чистого коду та чистої архітектури.

При розробці даного мобільного застосунку було використано фреймворк Flutter та мову програмування Dart. Розроблена архітектура має назву «трирівнева архітектура» та врахує усі принципи масштабованості та зручності. Для керування станом застосунку за допомогою Flutter було використано сучасні технології Cubit та Provider. Також було реалізовано багато функціональних можливостей для користувача, а саме: перегляд рекомендованих місць, визначення визначних місць за допомогою камери, створення та оновлення профілю, створення альбому фотографій, перегляд плану подорожі, ведення нотаток до плану, а також інтеграція з серверною частиною через API. Було забезпечено простий та зрозумілий інтерфейс для покращення взаємодії користувача з застосунком.

Також головним етап було проведення тестування мобільного застосунку, щоб забезпечити користувачеві безпеку, надійність та відповідність функціоналу його очікуванням. У процесі тестування перевірялися різні аспекти застосунку, включаючи його взаємодію з користувачем, відповідність дизайну та функціоналу вимогам, а також його продуктивність та стабільність під час різних умов використання. Результати тестування допомогли виявити та виправити будь-які

недоліки та помилки, забезпечивши якість та задоволення від користування застосунком.

Розроблений мобільний застосунок «Tripster» відповідає розробленій архітектурі та функціональним вимогам. Він забезпечує привабливий та зручний інтерфейс для планування подорожей визначними місцями. Кожен елемент інтерфейсу був уважно протестований на відповідність дизайну та його функціональному призначенню. Застосунок дає усі інструменти, що потрібні користувачу, для організації, збереження та пошуку інформації про визначні місця.

Проект має потенціал стати високоякісним та конкурентоспроможним рішенням на ринку мобільних застосунків для планування подорожей. Він відповідає сучасним стандартам розробки і використовує передові технології, такі як Flutter, Cubit та Provider, що сприяє створенню ефективного та зручного застосунку. Завдяки ретельному аналізу конкурентного середовища та врахуванню потреб користувачів, проект має можливість здобути популярність серед широкого кола користувачів та стати важливим інструментом для планування подорожей.

У висновку можна сказати, що мобільний застосунок успішно завершений та готовий для подальшого використання та розгортання. Його розробка відбувалася з урахуванням найсучасніших практик програмування та дизайну, що дозволило створити продукт відповідно до найвищих стандартів якості. Завдяки цьому, мобільний застосунок «Tripster» стане надійним інструментом для планування подорожей та здобуде популярність серед широкого кола користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Problems of Travel-Planning [Електронний ресурс]. URL: https://medium.com/@carlyjaddison/the-problems-of-travel-planning (Дата звернення 25.03.2024).

2. Benefits of Travel Mobile App Development [Електронний ресурс]. URL:https://www.oneclickitsolution.com/blog/benefits-of-travel-mobile-app-development (Дата звернення 25.03.2024).

3. What is Flutter? [Електронний ресурс]. URL: https://aws.amazon.com/what-is/flutter (Дата звернення 25.03.2024).

4. Flutter Evolution: How It's Shaping the Future of App Development [Електронний ресурс]. URL: https://www.dhiwise.com/post/flutter-evolution-how-it-shaping-the-future-of-app (Дата звернення 25.03.2024).

5. Three-tier Architecture with Flutter [Електронний ресурс]. URL: https://nimesha-dilini.medium.com/three-tier-architecture-with-flutterf8878dc9a6dc
(Дата звернення 10.04.2024)

6. What are cubits? [Електронний ресурс]. URL: https://dev.to/dev-vickie/fetching-apis-with-cubits-in-flutter-b8f (Дата звернення 15.04.2024)

7. Огляд архітектур управління станом на Flutter [Електронний ресурс]. URL: https://dou.ua/lenta/articles/flutter-architecture/ (Дата звернення 16.04.2024).

ДОДАТОК А

Структура проекту

```
|-- data
|   `-- repository
|       |-- auth_repository.dart
|       |-- gallery_repository.dart
|       |-- landmark_repository.dart
|       |-- place_repository.dart
|       |-- profile_repository.dart
|       `-- vacation_repository.dart
|-- domain
|   `-- models
|       |-- gallery_model.dart
|       |-- note_model.dart
|       |-- place_model.dart
|       |-- user_model.dart
|       |-- vacation_day_model.dart
|       `-- vacation_model.dart
|-- main.dart
|-- presentation
|   |-- cubits
|   |   |-- auth_cubit
|   |   |   |-- auth_cubit.dart
|   |   |   `-- auth_state.dart
|   |   |-- gallery_cubit
|   |   |   |-- gallery_cubit.dart
|   |   |   `-- gallery_state.dart
|   |   |-- landmark_cubit
|   |   |   |-- landmark_cubit.dart
|   |   |   `-- landmark_state.dart
|   |   |-- place_cubit
|   |   |   |-- place_cubit.dart
|   |   |   `-- place_state.dart
|   |   |-- profile_cubit
|   |   |   |-- profile_cubit.dart
|   |   |   `-- profile_state.dart
|   |   `-- vacation_cubit
|   |       |-- note_cubit.dart
|   |       |-- note_state.dart
|   |       |-- vacation_cubit.dart
|   |       `-- vacation_state.dart
|   |-- map_marker
|   |   `-- custom_map_marker.dart
|   |-- providers
|   |   `-- theme_providers.dart
|   |-- screens
|   |   |-- auth
|   |   |   |-- sign_in_screen.dart
|   |   |   `-- sign_up_screen.dart
|   |   |-- gallery
|   |   |   |-- add_collection_dialog.dart
|   |   |   |-- collection_photo_widget.dart
|   |   |   |-- detail_collection_widget.dart
|   |   |   `-- gallery_photo_screen.dart
```

```
|   |   |-- home
|   |   |   |-- detail_place_screen.dart
|   |   |   |-- home_screen.dart
|   |   |   |-- maps.dart
|   |   |   |-- place_list_widget.dart
|   |   |   `-- recommended_place_widget.dart
|   |   |-- landmark_recognation
|   |   |   `-- landmark_recognation.dart
|   |   |-- menu
|   |   |   `-- menu_screen.dart
|   |   |-- notes
|   |   |   |-- create_edit_note.dart
|   |   |   `-- note_screen.dart
|   |   |-- planning_trip
|   |   |   |-- card_widget.dart
|   |   |   |-- day_widget.dart
|   |   |   |-- detail_plan_screen.dart
|   |   |   |-- info_trip_widget.dart
|   |   |   |-- plan_trip_list_widget.dart
|   |   |   `-- plan_trip_screen.dart
|   |   |-- profile
|   |   |   |-- edit_profile_screen.dart
|   |   |   |-- my_profile_screen.dart
|   |   |   |-- user_info_widget.dart
|   |   |   `-- user_photo_widget.dart
|   |   `-- settings
|   |       |-- settings_buttons.dart
|   |       `-- settings_screen.dart
|   `-- widgets
|       |-- adress_widget.dart
|       |-- buttons
|       |   |-- change_theme_button.dart
|       |   `-- text_button.dart
|       |-- card
|       |   |-- photo_card.dart
|       |   `-- place_card.dart
|       |-- custom_search_widget.dart
|       |-- custom_text_widget.dart
|       |-- custom_textfield.dart
|       |-- headers
|       |   |-- custom_appbar.dart
|       |   |-- header_widget.dart
|       |   `-- place_header_widget.dart
|       `-- snack_bar_widget.dart
`-- utils
    |-- constants.dart
    |-- dark_theme
    |   `-- dark_theme.dart
    |-- languages
    |   `-- generated
    |       |-- codegen_loader.g.dart
    |       `-- locale_keys.g.dart
    |-- location_to_adress.dart
    |-- shared_pref.dart
    `-- theme.dart
```

ДОДАТОК Б

Код методу

```
Future<Landmark> getLandmark(XFile pickedFile) async {
    final imageBytes = await pickedFile.readAsBytes();
    final response = await http.post(
      Uri.parse(
          'https://vision.googleapis.com/v1/images:annotate?key=KEY),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode({
        'requests': [
          {
            'image': {'content': base64Encode(imageBytes)},
            'features': [
              {'type': 'LANDMARK_DETECTION'}
            ]
          }
        ]
      }),
    );
    if (response.statusCode == 200) {
      final responseData = jsonDecode(response.body);
      print(responseData);
      final landmarkAnnotations =
          responseData['responses']?[0]?['landmarkAnnotations'];

     if (landmarkAnnotations != null &&
landmarkAnnotations.isNotEmpty) {
        final landmark = landmarkAnnotations[0];
        return Landmark(
          description: landmark['description'],
          image: File(pickedFile.path),
        );
      } else {
        return Landmark(description: 'Landmark not found', image:
null);
      }
    } else {
      throw Exception('Request failed with status:
${response.statusCode}');
    }
  }
}
```

ДОДАТОК В

Участь у конференції

Рисунок В.1 – Сертифікат про участь у конференції

ПЛАНУВАННЯ ТА ОРГАНІЗАЦІЯ ПОДОРОЖЕЙ ЗА ДОПОМОГОЮ МОБІЛЬНОГО ЗАСТОСУНКУ НА FLUTTER

Вельма Анастасія Олександрівна
Харківський національний університет радіоелектроніки
студент кафедри Програмної інженерії
Онищенко Костянтин Георгійович
Харківський національний університет радіоелектроніки
Старший викладач кафедри Програмної інженерії
м. Харків, Україна

Ключові Слова: Застосунок, Flutter, Подорожі

В наш час багато людей подорожують світом, відкриваючи нові країни, культури та пригоди. Сфера подорожей та організації поїздок стає все більш популярною в світі. Люди прагнуть знайомитися з культурами інших країн, відвідувати нові визначні місця, а також створювати незабутні спогади. Однак планування подорожі та її організація є складним процесом, що займає багато часу та зусиль [1], тому ефективна організація подорожей стає найважливішим завданням у цьому процесі. Саме тому виникає потреба у зручному інструменті, який би допомагав користувачам ефективно організовувати свої подорожі [2]. Розробка мобільного застосунку за допомогою фреймворка Flutter значно спростить процес планування подорожі. Застосунок допоможе мандрівникам забезпечити зручність на кожному етапі їх поїздки.

Цей мобільний застосунок призначений для планування та організації подорожей. Він має полегшити процес планування місць, які користувач хоче відвідати, формування бюджету конкретної поїдки, а також збирати усю інформацію з поїздки (фотографії, відвідувані місця тощо) для подальшого використання. Мета проекту – створити зручний і функціональний мобільний застосунок, який буде корисним для мандрівників усього світу, що прагнуть ефективно планувати та організовувати власні подорожі.

На ринку вже існують кілька мобільних застосунків, що виконують схожі

Рисунок В.2 – Текст тези

функції, а саме: TravelSpend, Expedia та TripIt. Проте більшість з них мають обмежений функціонал або не враховують усі аспекти планування та організації поїздки. Крім того, вони часто є платними або містять багато недоречної реклами, що є незручним для користувачів.

Для розробки мобільного застосунку було обрано кросплатформлену технологію Flutter від Google [3]. Застосунки, розроблені з Flutter, відрізняються високою швидкістю роботи та плавною анімацією [4]. Це робить їх комфортними та простими у використанні. Використання Flutter дозволить створити високоякісний, зручний та доступний для широкого кола користувачів мобільний застосунок.

Застосунок матиме такі основні функції:

– створення фотоальбомів та нотаток для збереження спогадів про подорож;

– планування маршруту та визначних пам’яток для відвідування з використанням карт та рекомендацій;

– маркування відвіданих місць на карті;

– розпізнавання визначних місць за допомогою камери;

– рекомендації для користувача щодо місць для відвідування.

Для розпізнавання визначних місць за допомогою камери буде використано Landmark recognition від Firebase. Ця технологія базується на використанні штучного інтелекту та машинного навчання для ідентифікації відомих місць на основі зображень [5]. За допомогою Landmark recognition, користувач зможе визначити інформацію про це місце, наприклад, його назву, адресу, тип (парк, музей, історична пам’ятка тощо), опис тощо. Ця функція значно полегшить користувачам процес вивчення нових місць та забезпечить їм доступ до цікавої та корисної інформації прямо на момент відвідування.

У результаті можна сказати, що розробка мобільного застосунку для планування та організації подорожей за допомогою Flutter відповідає потребам сучасного суспільства. Такий застосунок спросить процес планування та забезпечить користувачам зручний інструмент для створення незабутніх

Рисунок В.3 – Продовження тексту тези

подорожей. Використання інноваційних технологій, таких як Landmark recognition підвищить зручність та функціональність у використанні. Використання кросплатформленої технології Flutter дозволить забезпечити швидку роботу та плавну анімацію застосунку, що зробить його комфортним та привабливим для користувачів.

Розробка мобільного застосунку для планування подорожей на базі Flutter з Landmark recognition від Firebase має потенціал стати конкурентоспроможною альтернативою існуючим рішенням на ринку, а також стати популярним інструментом для мандрівників усього світу, які прагнуть ефективно організовувати свої поїздки та зберігати незабутні спогади.

СПИСОК ДЖЕРЕЛ

1. The Problems of Travel-Planning [Електронний ресурс]. URL: https://medium.com/@carlyjaddison/the-problems-of-travel-planning (Дата звернення 25.03.2024).

2. Benefits of Travel Mobile App Development [Електронний ресурс]. URL: https://www.oneclickitsolution.com/blog/benefits-of-travel-mobile-app-development (Дата звернення 25.03.2024).

3. What is Flutter? [Електронний ресурс]. URL: https://aws.amazon.com/what-is/flutter (Дата звернення 25.03.2024).

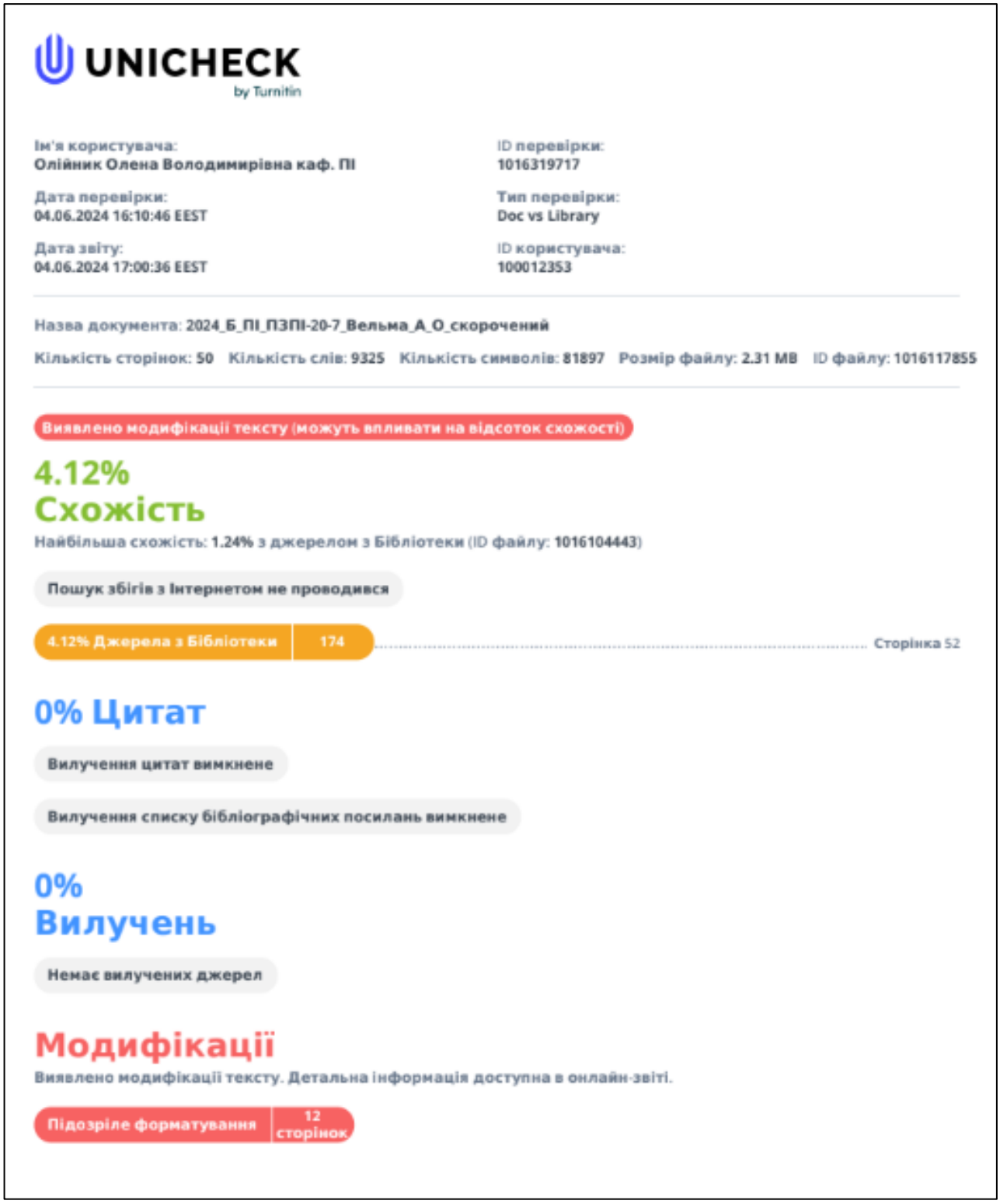
4. Flutter Evolution: How It's Shaping the Future of App Development [Електронний ресурс]. URL: https://www.dhiwise.com/post/flutter-evolution-how-it-shaping-the-future-of-app (Дата звернення 25.03.2024).

5. Landmark Recognition. [Електронний ресурс]. URL: https://firebase.google.com/docs/ml/recognize-landmarks (Дата звернення 25.03.2024).

Рисунок В.3 – Закінчення тексту тези

ДОДАТОК Г

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016319717

Дата перевірки:
04.06.2024 16:10:46 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
04.06.2024 17:00:36 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-7_Вельма_А_О_скорочений

Кількість сторінок: 50 Кількість слів: 9325 Кількість символів: 81897 Розмір файлу: 2.31 MB ID файлу: 1016117855

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.12%
Схожість

Найбільша схожість: 1.24% з джерелом з Бібліотеки (ID файлу: 1016104443)

Пошук збігів з Інтернетом не проводився

4.12% Джерела з Бібліотеки 174 ..... Сторінка 52

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

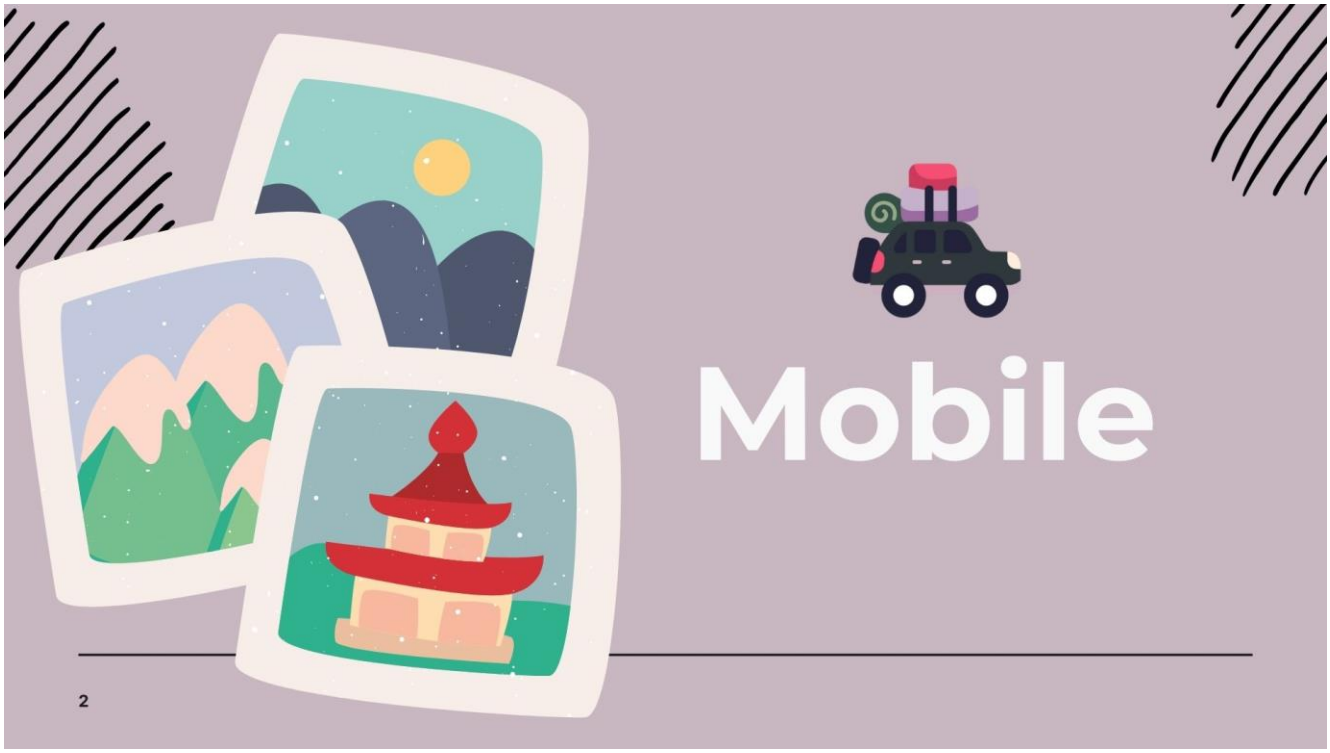
Немає вилучених джерел

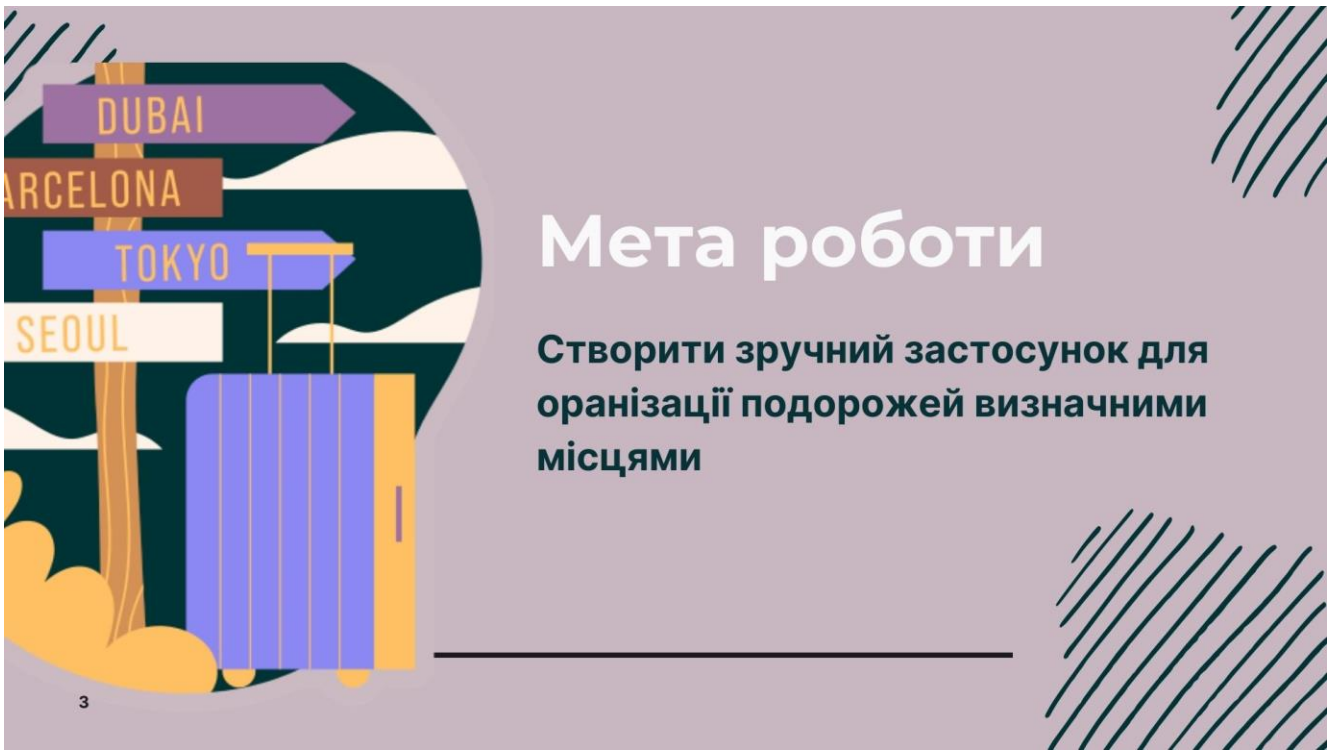
Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 12 сторінок

ДОДАТОК Д
Слайди презентації




DUBAI
ARCELONA
TOKYO
SEOUL
Мета роботи
Створити зручний застосунок для оранізації подорожей визначними місцями
3


Аналоги
• TravelSpend
• Expedia
• TripIt
4

В чому потреба?

- Комплексний підхід до планування подорожі
- Визначення найкращих варіантів для мандрівки
- Створення фото-колекцій зі створених подорожей
- Швидке розпізнавання місць за допомогою камери
- Можливість кастомізувати інтерфейс під власні потреби

5

Постановка задачі | Опис системи

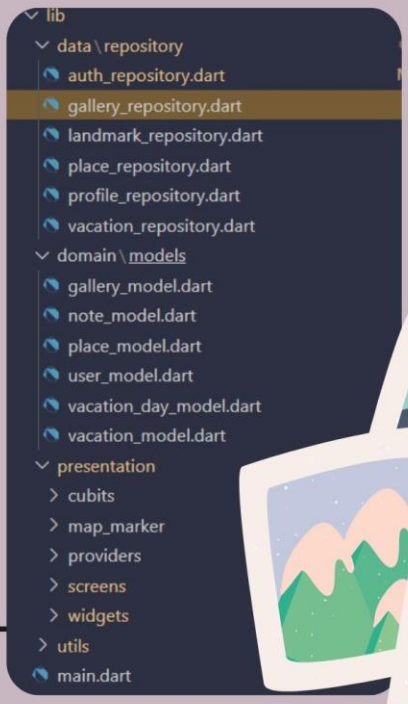
6

Технології розробки

- Flutter
- Google Vision API
- Google Map
- Cubit
- shared_preferences
- easy_localizations

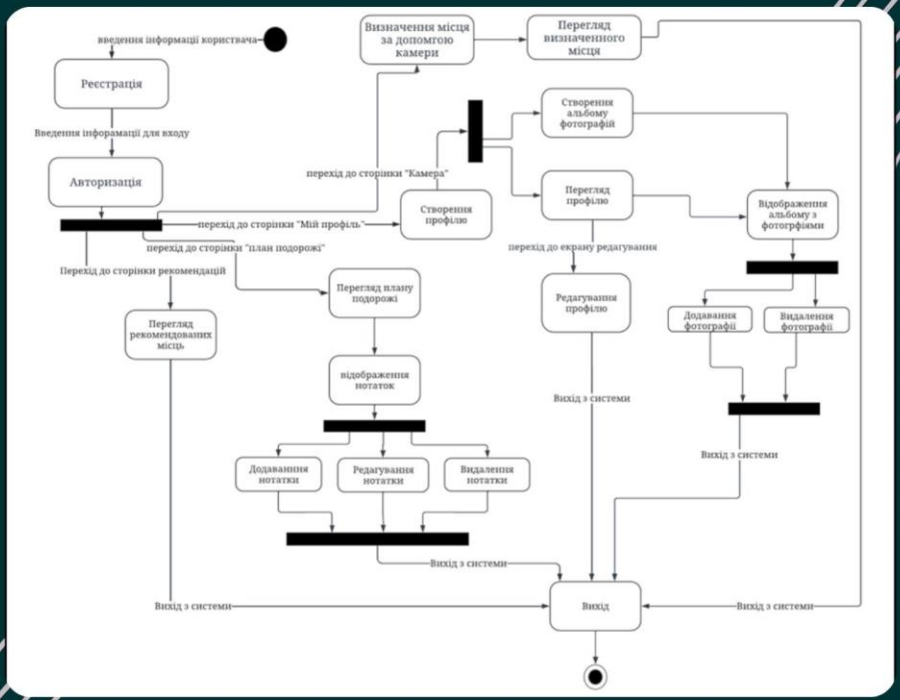
7

Трьохрівнева архітектура

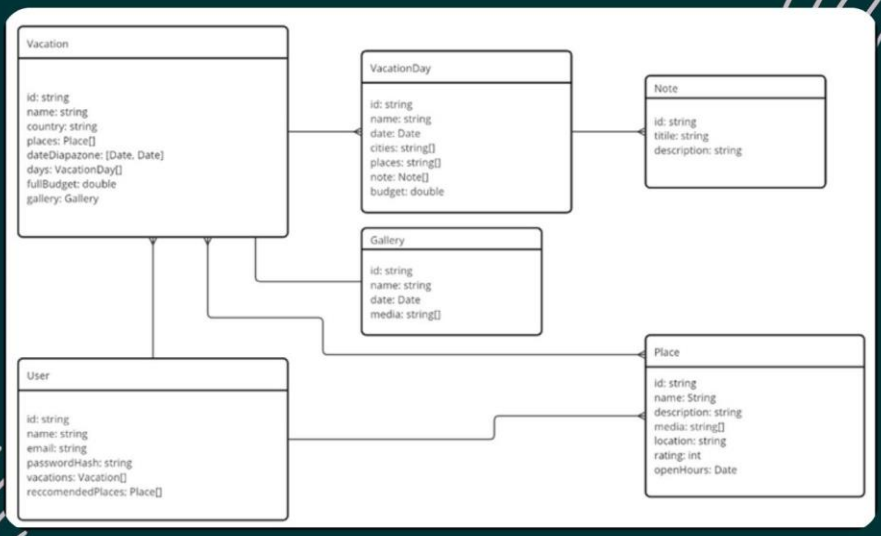


8

Діаграма діяльності

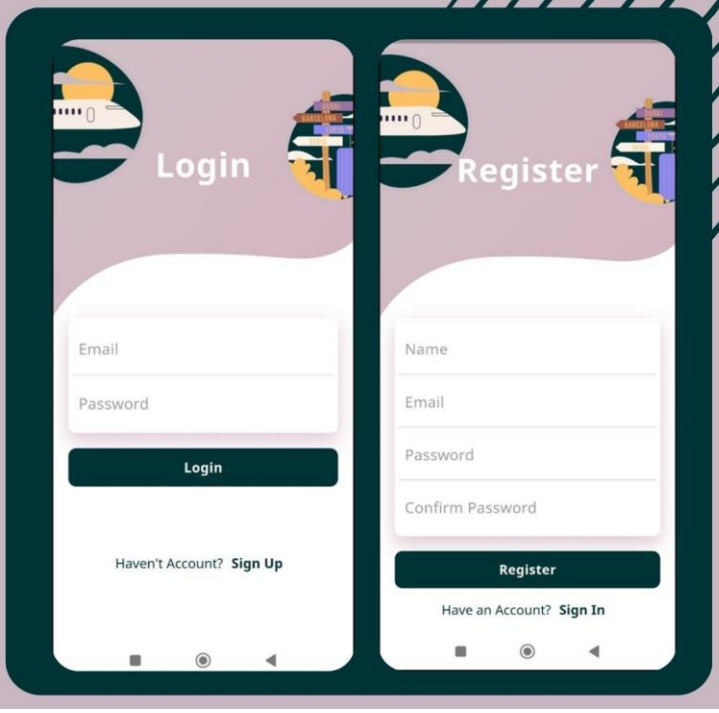


Діаграма класів



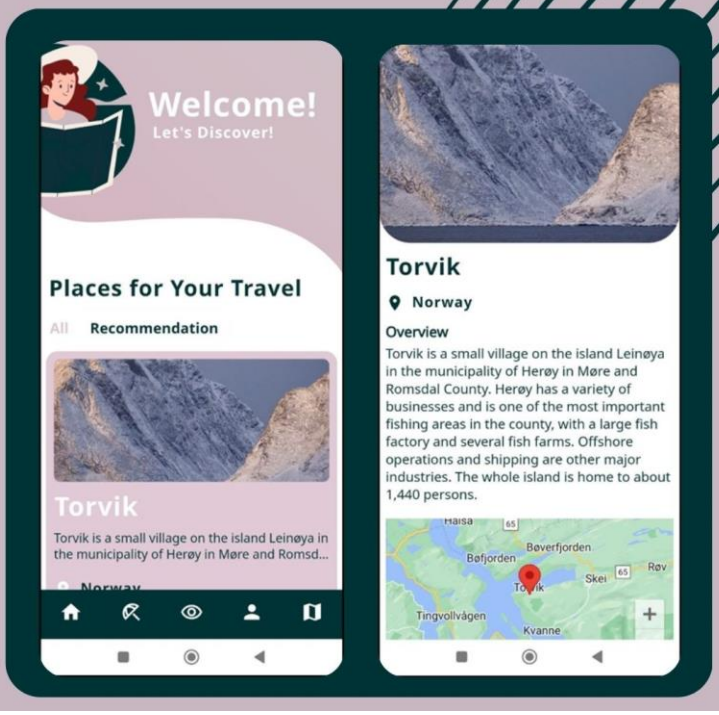
Авторизація

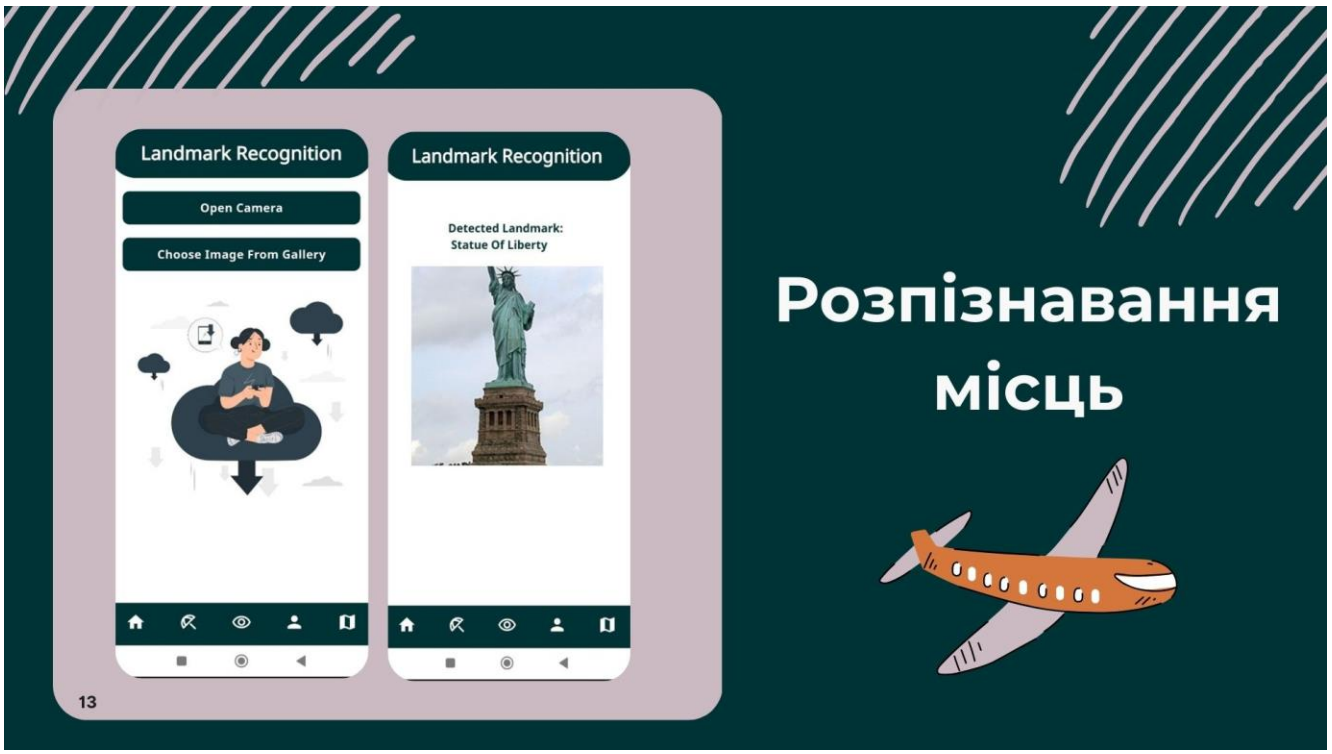
11

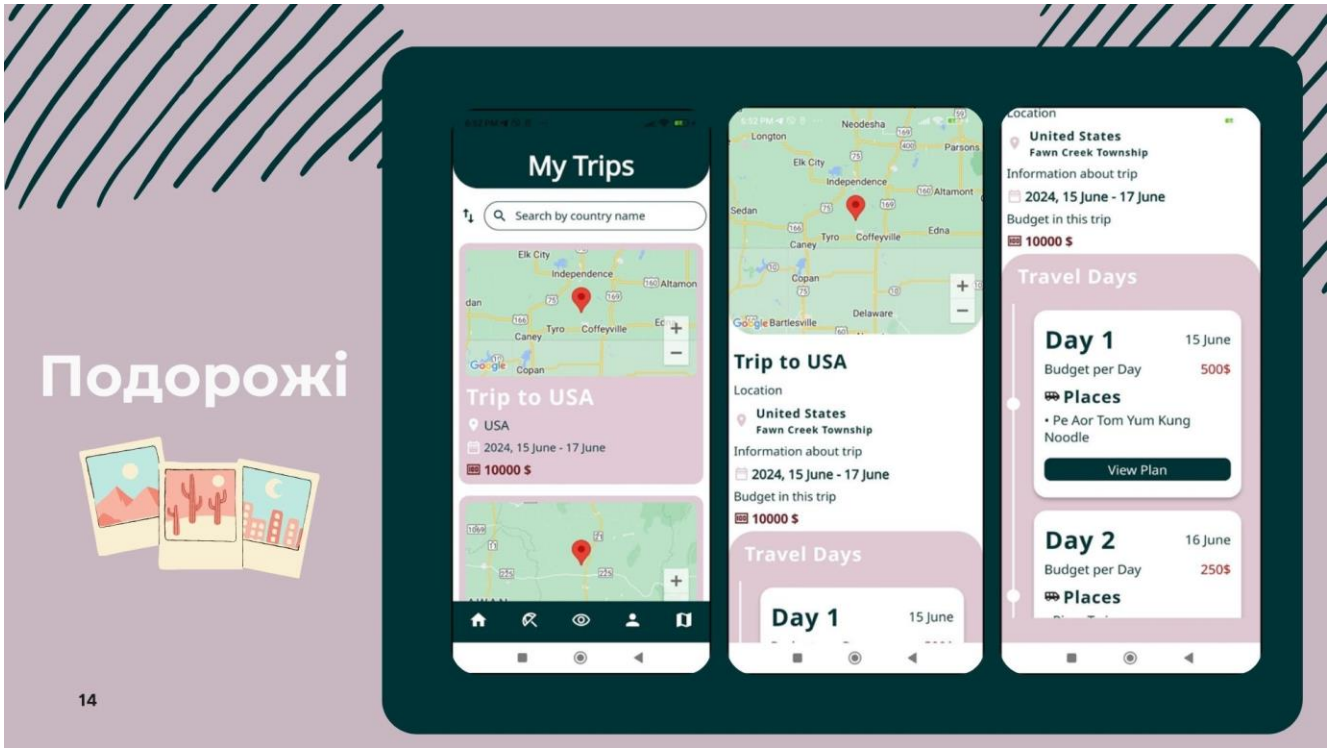


Рекомендовані та цікаві місця

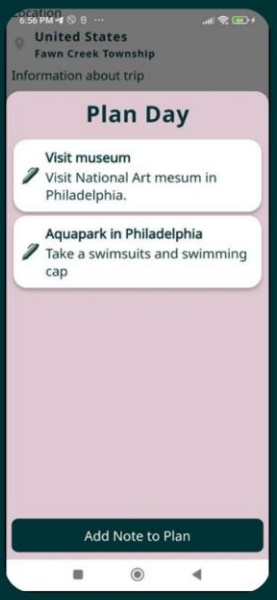
12




Landmark Recognition
Open Camera
Choose Image From Gallery
Landmark Recognition
Detected Landmark:
Statue Of Liberty
Розпізнавання місць
13


Подорожі
My Trips
Search by country name
Trip to USA
USA
2024, 15 June - 17 June
10000 $
Trip to USA
Location
United States
Fawn Creek Township
Information about trip
2024, 15 June - 17 June
Budget in this trip
10000 $
Travel Days
Day 1
15 June
Budget per Day
500$
Places
• Pe Aor Tom Yum Kung Noodle
View Plan
Day 2
16 June
Budget per Day
250$
Places
14

Планування дня у подорожі



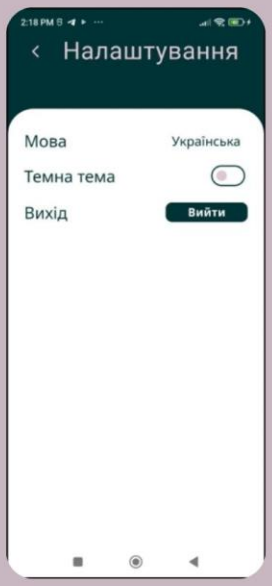
15

Профіль користувача

16

Налаштування

17



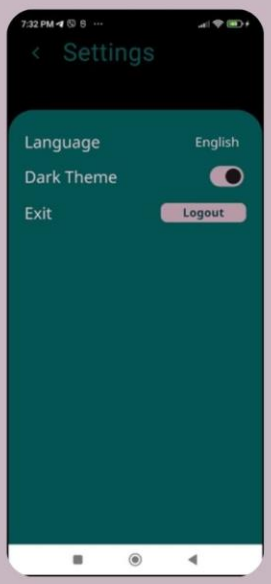
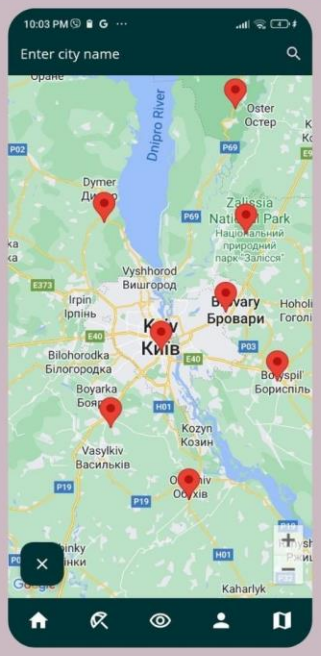


Фото-колекції

18

Карта з маркуваннями



19

Приклад тест-кейсу на створення нового користувача

20

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Опис функції:	Реєстрація нового користувача		
Власник тесту:	Вельма Анастасія Олександрівна		
Дата створення:	01.06.2024		
Мета тесту:	Перевірка процесу реєстрації нового користувача у застосунку		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити завантажений застосунок	Користувач має доступ до застосунку	Пройдено
Реєстрація нового користувача			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити застосунок «Tripster»	Застосунок відкривається і відображається головний екран з кнопкою «Реєстрація».	Пройдено
2	Натиснути на кнопку «Реєстрація».	Відкривається форма реєстрації з полями для введення даних.	Пройдено
3	Заповнити обов'язкове поле «Ім'я»	Поле заповнене, кількість символів правильна, значення збережене.	Пройдено
4	Заповнити обов'язкове поле «Пошта»	Поле заповнене та пройшло валідацію на коректність (правильний формат пошти).	Пройдено
4	Заповнити обов'язкове поле «Пароль»	Поле заповнене, пароль відповідає вимогам безпеки (не менше 8 символів, містить літери та цифри)	Пройдено
5	Заповнити обов'язкове поле «Підтвердити пароль».	Поле заповнене, введений пароль збігається з паролем у попередньому полі.	Пройдено
6	Натиснути кнопку «Зареєструватися»	Застосунок відправляє дані на сервер для реєстрації нового користувача. Після успішної реєстрації, користувач автоматично входить у систему.	Пройдено
Результати тестування			
Тестувальник: Вельма А.О.	Дата прогону тесту: 01.06.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

Висновок

У висновку можна сказати, що було створено зручний застосунок для організації мандрівок. Застосунок надає користувачу багато можливостей, а саме: перегляд цікавих та рекомендованих місць, створення та редагування власного профілю, створення фото-колецій, кастомізація застосунку під власні потреби, перегляд подорожі та планування дня, розпізнавання місць та перегляд карт, що робить його незамінним помічником для будь-якого мандрівника.

21

Публікація

ПЛАНУВАННЯ ТА ОРГАНІЗАЦІЯ ПОДОРОЖЕЙ ЗА ДОПОМОГОЮ МОБІЛЬНОГО ЗАСТОСУНКУ НА FLUTTER

Вельма Анастасія Олександрівна
Харківський національний університет радіоелектроніки
студент кафедри Програмної інженерії
Онищенко Костянтин Георгійович
Харківський національний університет радіоелектроніки
Старший викладач кафедри Програмної інженерії
м. Харків, Україна

Ключові Слова: Застосунок, Flutter, Подорожі

В наш час багато людей подорожують світом, відкриваючи нові країни, культури та пригоди. Сфера подорожей та організації поїздок стає все більш популярною в світі. Люди прагнуть знайомитися з культурами інших країн, відвідувати нові визначні місця, а також створювати незабутні спогади. Однак планування подорожі та її організація є складним процесом, що займає багато часу та зусиль [1], тому ефективна організація подорожей стає найважливішим завданням у цьому процесі. Саме тому виникає потреба у зручному інструменті, який би допомагав користувачам ефективно організовувати свої подорожі [2]. Розробка мобільного застосунку за допомогою фреймворка Flutter значно спростить процес планування подорожі. Застосунок допоможе мандрівникам забезпечити зручність на кожному етапі їх поїздки.

Цей мобільний застосунок призначений для планування та організації подорожей. Він має полегшити процес планування місць, які користувач хоче відвідати, формування бюджету конкретної поїздки, а також збирати усю інформацію з поїздки (фотографії, відвідувані місця тощо) для подальшого використання. Мета проекту – створити зручний і функціональний мобільний застосунок, який буде корисним для мандрівників усього світу, що прагнуть ефективно планувати та організовувати власні подорожі.

На ринку вже існують кілька мобільних застосунків, що виконують схожі

260

У результаті можна сказати, що розробка мобільного застосунку для планування та організації подорожей за допомогою Flutter відповідає потребам сучасного суспільства. Такий застосунок спросить процес планування та забезпечить користувачам зручний інструмент для створення незабутніх

261

262

22

Дякую за увагу!