Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування»**

Виконав: студент 2-го курсу, групи ЗПІ-23-1
спеціальності 121 – Інженерія програмного забезпечення
(шифр та назва напряму підготовки, спеціальності)
_____ Дела В.П.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ:
_____ Войтко В.В.
(прізвище та ініціали)
«08» грудня 2023 р.

Опонент:
к.т.н., доц. каф. ОТ Савицька Л.А.
(прізвище та ініціали)
«08» грудня 2023 р.

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н, проф. Романюк О.Н.
(прізвище та ініціали)
«08» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

# З А В Д А Н Н Я
# НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙУ РОБОТУ СТУДЕНТУ

Деді Владиславу Петровичу

1. Тема роботи – розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 року

3. Вихідні дані до роботи: середовище розробки Android Studio, мова розробки Kotlin, система керування базами даних – Firebase, операційна система – Windows 10, метод моніторингу якості обслуговування, метод пошуку закладів харчування.

4. Зміст текстової частини: вступ; аналіз стану розвитку мобільних систем для бронювання місць та постановка задач дослідження; розробка методу та моделей Android-системи; розробка Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування, тестування програми; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: графічний інтерфейс мобільного застосунку; модель роботи мобільного застосунку; блок-схеми алгоритмів роботи мобільного застосунку; тестування мобільного застосунку.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Войтко В. В., к.т.н., доцент кафедри ПЗ | 19.09.23 | 05.12.23 |
| 5 | Причепа І.В, к.е.н., доцент кафедри ЕПВМ | 22.11.23 | 01.12.23 |

7. Дата видачі завдання 19 вересня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз стану розвитку мобільних систем для бронювання місць та постановка задач дослідження | 20.09.2023 30.09.2023 | Вик. |
| 2 | Розробка методів та моделей Android-системи | 01.10.2023 17.10.2023 | Вик |
| 3 | Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування | 18.10.2023 04.11.2023 | Вик |
| 4 | Тестування програми | 05.11.2023 21.11.2023 | Вик. |
| 5 | Економічна частина | 22.11.2023 01.12.2023 | Вик. |

Студент _____ Дела В.П
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Войтко В.В.
(підпис) (прізвище та ініціали)

# Анотація

УДК 004.912.032.26

Деда В.П. Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 161с.

На укр. мові. Бібліогр.: назв: 51; рисунків: 54; таблиць: 12.

У магістерській кваліфікаційній роботі подано результати дослідження технологій розробки Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування. Обґрунтовано доцільність удосконалення методів та засобів для пошуку закладів за заданими критеріями.

Магістерська кваліфікаційна робота містить аналіз відомих додатків для бронювання місць у закладах харчування з можливістю залишати відгуки та оцінювати якість обслуговування, методів пошуку закладів за запитами користувачів.

Розроблено метод пошуку закладів під потреби та бажання користувача. Розроблено модуль для додавання закладів харчування у систему власникам цих закладів, пошук закладів за містом та тегами, а також забезпечено можливість бронювання місць і можливість оцінити обслуговування в закладі. Розроблено Android-систему «RestoBooking» з використанням інтегрованого середовища розробки Android Studio, мови програмування Kotlin та Jetpack Compose для побудови графічного інтерфейсу користувача. Список закладів зберігається на Cloud Firestore, фотографії закладів зберігаються на Firebase Storage, а авторизація користувачів відбувається через Firebase Auth.

Ключові слова: Android-система, бронювання закладів, оцінювання якості обслуговування.

## Abstract

Deda V.P Development of methods and software tools of the Android system for booking seats and monitoring the quality of service in catering establishments. Master's thesis on the specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. Number of pages 161 p.

In Ukrainian speech Bibliographer: titles;51, drawings:54; tables:12.

The master's qualification thesis presents the results of the research of technologies for the development of the Android system for seat reservations and monitoring the quality of service in catering establishments. The expediency of improving the methods and tools for searching for institutions according to the given criteria is substantiated.

The master's thesis includes an analysis of well-known applications for booking places in food establishments with the ability to leave feedback and evaluate the quality of service, methods of searching for establishments according to the user's needs and wishes.

A method of searching for institutions has been developed according to the user's needs and wishes, so that it is easier to find the desired institution. A module has been developed for adding restaurants to the system for the owners of these establishments, searching for establishments by city and tags, as well as the possibility of booking seats and the ability to rate the service at the establishment. The Android system "RestoBooking" was developed using the Android Studio integrated development environment, the Kotlin programming language and Jetpack Compose for building a graphical user interface. The list of venues is stored on Cloud Firestore, photos of venues are stored on Firebase Storage, and user authorization is via Firebase Auth.



Keywords: Android system, reservation of facilities, assessment of service quality.

ЗМІСТ

ВСТУП ..... 5

1 АНАЛІЗ СТАНУ РОЗВИТКУ ANDROID СИСТЕМ БРОНЮВАННЯ МІСЦЬ І МОНІТОРИНГУ ЯКОСТІ ОБСЛУГОВУВАННЯ В ЗАКЛАДАХ ХАРЧУВАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ ..... 9

1.1 Аналіз стану розвитку Android систем бронювання місць і моніторингу якості обслуговування в закладах харчування ..... 9

1.2 Порівняння з аналогами ..... 11

1.3 Аналіз методів розв'язання задачі ..... 19

1.4 Постановка задач дослідження ..... 23

1.5 Висновки ..... 24

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ СИСТЕМИ ..... 25

2.1. Архітектурне проєктування програмної системи ..... 25

2.2 Розробка методу реєстрації нового закладу ..... 29

2.3 Розробка методу пошуку закладів харчування ..... 32

2.4 Розробка методу моніторингу якості обслуговування ..... 35

2.5 Розробка методу бронювання місць у закладах харчування ..... 39

2.6 Розробка моделі Android-системи для бронювання і моніторингу якості обслуговування в закладах харчування ..... 41

2.7 Висновки ..... 44

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ANDROID-СИСТЕМИ ДЛЯ БРОНЮВАННЯ МІСЦЬ ТА МОНІТОРИНГУ ЯКОСТІ ОБСЛУГОВУВАННЯ В ЗАКЛАДАХ ХАРЧУВАННЯ ..... 45

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного продукту ..... 45

3.2 Розробка програмного модуля реєстрації користувача ..... 48

3.3 Розробка програмного модуля реєстрації нового закладу харчування ..... 50
3.4 Розробка програмного модуля додавання коментарів ..... 52
3.5 Розробка програмного модуля завантаження усіх закладів харчування ..... 54
3.6 Розробка програмного модуля для завантаження закладу харчування за ID ..... 56
3.7 Розробка програмного модуля для бронювання місць у закладах харчування ..... 59
3.8 Висновки ..... 61
4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ..... 62
4.1 Аналіз методів тестування ..... 62
4.2 Тестування Android-системи для бронювання місць і моніторингу якості обслуговування у закладах харчування ..... 65
4.3 Розробка інструкції користувача ..... 81
5 ЕКОНОМІЧНА ЧАСТИНА ..... 83
5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки ..... 83
5.2 Розрахунок витрат на проведення науково-дослідної роботи ..... 87
5.2.1 Витрати на оплату праці ..... 87
5.2.2 Відрахування на соціальні заходи ..... 90
5.2.3 Сировина та матеріали ..... 90
5.2.4 Розрахунок витрат на комплектуючі ..... 91
5.2.5 Спецустаткування для наукових (експериментальних) робіт ..... 91
5.2.6 Програмне забезпечення для наукових (експериментальних) робіт ..... 92
5.2.7 Амортизація обладнання, програмних засобів та приміщень ..... 93
5.2.8 Паливо та енергія для науково-виробничих цілей ..... 94
5.2.9 Службові відрядження ..... 95
5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації ..... 96

5.2.11 Інші витрати ..... 96

5.2.12 Накладні (загальновиробничі) витрати ..... 96

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором ..... 97

5.4 Висновки ..... 102

ВИСНОВКИ ..... 103

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 104

ДОДАТКИ ..... 108

Додаток А – Технічне завдання ..... 109

Додаток Б – Протокол перевірки на плагіат ..... 114

Додаток В – Лістинг програми ..... 115

Додаток Г – Ілюстративна частина ..... 148

# ВСТУП

**Обґрунтування вибору теми дослідження**. Ситуація, що склалася в ресторанному бізнесі, вимагає від компаній переходу на новий рівень роботи та обслуговування клієнтів. Через складне економічне середовище та зростаючу конкуренцію традиційні методи ведення бізнесу вже не гарантують підвищення прибутковості, а іноді навіть ускладнюють утримання досягнутого рівня. Така ситуація вимагає від власників закладів харчування аналізу сучасних світових тенденцій та впровадження сучасних методів і підходів до ведення бізнесу. У цьому контексті одним із найефективніших способів трансформації бізнес-процесів є діджиталізація, яка має на меті оцифрування окремих процесів [1].

Використання нових технологій у наданні послуг закладами харчування вже давно стоїть на порядку денному в багатьох країнах. Згідно з опитуванням, проведеним у Франції в 2014 році, 63% респондентів використовували нові технології при виборі ресторану або кафе, а також вони використовували нові технології для розуміння меню, вибору їжі, замовлення столиків і обідів онлайн, а також для оплати послуг в електронному вигляді [2].

Повний перегляд принципів роботи дозволив багатьом підприємцям вижити і навіть процвітати. Індустрія ресторанного бізнесу більше не могла ігнорувати вимоги часу і була змушена впроваджувати нові технології. Цю індустрію довелося переоцінити з огляду на те, що розвиток бізнесу визначається не тільки хорошим чи поганим сервісом, тепер він залежить від того, як відбувається взаємодія з рестораном за його межами. Серед функцій, які є важливими для користувачів сьогодні, виділимо:

- онлайн замовлення і доставку;
- безконтактну оплату;
- онлайн-бронювання столиків, що дозволяє збирати дані про клієнтів та їхні смакові вподобання, щоб впливати на політику лояльності закладу;
- доступ до онлайн-меню та замовлень через QR-коди.

Більшість мобільних додатків для бронювання реалізовані так, що бронювання місць у закладах харчування доступне лише в одному чи визначених кількох містах. Тому розробка Android-системи для бронювання місць і моніторингу якості обслуговування, що не прив'язується до певної локації і дає можливість зареєструвати заклад у будь-якому місті, а також дозволяє зручно і швидко знайти заклад завдяки реалізації спеціальних тегів, є досить актуальною задачею на сьогодні.

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

**Мета та завдання дослідження**. Метою магістерської кваліфікаційної роботи є підвищенням ефективності пошуку і бронювання місць у закладах харчування шляхом розробки спеціалізованого Android застосунку з удосконаленням методів і засобів пошуку закладів, що дозволить пришвидшити пошукові процеси та сприятиме розвитку комунікацій з клієнтами в інтерактивному режимі.

Основними задачами роботи є:

- визначити засоби реалізації мобільної системи бронювання місць і моніторингу якості обслуговування в закладах харчування;
- розробити метод додавання нового закладу;
- розробити метод пошуку закладів харчування;
- розробити метод моніторингу якості обслуговування;
- розробити метод бронювання місць у закладах харчування;
- розробити усі програмні модулі компонентів системи мобільного застосунку;
- провести тестування розробленої системи.

**Об'єктом дослідження** є процес розробки Android-системи для бронювання місць і моніторингу якості обслуговування у закладах харчування.

**Предметом дослідження** є методи та засоби розробки спеціалізованої

Android-системи.

**Методи дослідження.** У процесі досліджень використовувались методи дослідження:

- методи розробки мобільного застосунку під операційну систему Android для реалізації мобільної системи бронювання місць і моніторингу якості обслуговування у закладах харчування;
- методи об’єктно-орієнтованого програмування та шаблони програмування для розробки архітектури класів і сервісів програмного забезпечення;
- методи створення дизайну для розробки інтерфейсу мобільного застосунку;
- методи тестування для перевірки працездатності мобільного застосунку.

**Наукова новизна отриманих результатів**:

- подальшого розвитку отримав метод пошуку закладів харчування для бронювання місць, який, на відміну від існуючих, має персоналізований підхід до пошуку та застосовує адаптивні алгоритми з індивідуалізацією пошукових процесів, що дозволяє покращити інтерактивні комунікації з клієнтами;
- подальшого розвитку отримав метод моніторингу якості обслуговування в закладах харчування, який, на відміну від існуючих, реалізує в середовищі мобільної системи спеціалізовані засоби підтримки інтерактивної взаємодії працівників з клієнтами, що впливає на визначення статусу закладу харчування в системі рейтингового оцінювання в процесі формування списку пріоритетних варіантів і сприяє удосконаленню бізнес-стратегії закладів харчування.

**Практичне значення одержаних результатів.** Розроблена Android-система може використовуватися закладами харчування для пришвидшення процесу бронювання місць, а також для отримання зворотного зв’язку про якість обслуговування.

**Особистий внесок здобувача.** Усі наукові результати, викладені у

магістерській кваліфікаційній роботі, отримані автором особисто. У науковій роботі, опублікованій у співавторстві, автору належать такі результати: аналіз функціоналу мобільного додатку, постановка задач дослідження, визначення методів їх розв’язання і програмна реалізація системи.

**Апробація матеріалів магістерської кваліфікаційної роботи.** Результати магістерської кваліфікаційної роботи обговорювалися на міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023).

**Публікації.** Результати дослідження опубліковані в тезах доповіді міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ, 2023»[3].

**Структура та обсяг роботи.** Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел, що містить 35 найменувань, 4 додатки. Робота містить 54 ілюстрацій, 12 таблиць.

# 1 АНАЛІЗ СТАНУ РОЗВИТКУ ANDROID СИСТЕМ БРОНЮВАННЯ МІСЦЬ І МОНІТОРИНГУ ЯКОСТІ ОБСЛУГОВУВАННЯ В ЗАКЛАДАХ ХАРЧУВАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Аналіз стану розвитку Android систем бронювання місць і моніторингу якості обслуговування в закладах харчування

Мобільні технології є не тільки актуальними, але й важливою частиною сучасного життя, впливаючи на спосіб спілкування, праці, навчання, відпочинку. Їхній стрімкий розвиток відкриває нові горизонти для інновацій та можливостей у різних галузях.

Система Android, розроблена компанією Google, є однією з найпопулярніших операційних систем для мобільних пристроїв на сьогоднішній день [4]. Її популярність ґрунтується на кількох ключових факторах:

- Android має велику користувацьку базу по всьому світу, мільярди смартфонів і планшетів працюють на цій операційній системі;
- операційна система Android працює на пристроях різних виробників і моделей, що дає користувачам широкий вибір від доступних смартфонів до флагманських моделей, які підходять для різних бюджетів і потреб;
- Android базується на відкритому джерелі коду, що сприяє широкій спільноті розробників;
- система Android пропонує різноманітні функції, включаючи розширену підтримку для розширеної реальності (AR), інтернету речей (IoT), віртуальної реальності (VR), голосового керування та інших інноваційних технологій;
- Google постійно вдосконалює Android, випускаючи нові версії з покращеннями щодо продуктивності, безпеки та функціональності.

Мобільні додатки для бронювання житла, транспорту і багатьох інших місць є досить актуальними та займають велику частину серед мобільних додатків. Вони надають можливість зручно і швидко здійснити бронювання

обраного варіанту. Такі системи покращують організацію обслуговування та зменшують ризик недоречних затримок та конфліктів.

Окрім бронювання місць, сучасні Android-додатки пропонують додатковий функціонал, який робить їх ще більш привабливими для клієнтів і власників ресторанів. Система коментарів є важливим й актуальним функціоналом для такого типу додатків. Користувачі можуть додавати коментарі та відгуки про свій візит, а інші клієнти можуть користуватися цією інформацією для вибору місця для обіду чи вечері [5].

Ця інформація стає доступною для всіх користувачів, вона допомагає іншим клієнтам обирати місця з вищими рейтингами та позитивними відгуками. Це дозволяє підвищити рівень обслуговування, адже власники закладів зацікавлені у високих рейтингах і хороших відгуках, оскільки від цього прямо залежить репутація та прибуток закладу.

Підбір закладів під потреби користувача є важливим аспектом, оскільки він сприяє задоволенню індивідуальних вимог та очікувань відвідувачів. Кожен має свої власні смаки, уподобання і обмеження, і вибір правильного закладу може значно покращити емоції від відвідування ресторану, кафе чи іншого закладу харчування.

Підбір належного закладу допомагає користувачам знайти атмосферу та стиль, які відповідають їхнім особистим уподобанням та намірам, отримати задоволення від якісного обслуговування та приємної атмосфери під час візиту, а також зменшити ризик розчарування через неправильний вибір закладу. Тому підбір правильного закладу, який враховує індивідуальні потреби та уподобання, робить відвідування ресторанів та кафе більш задовільним і приємним досвідом для кожного користувача.

Отже, Android-система для бронювання місць і моніторингу якості обслуговування в закладах харчування є досить зручним інструментом для управління такими закладами, а також для бачення реального стану обслуговування та рівня задоволення користувачів від відвідування закладу.

Зважаючи на високий темп діджиталізації сфери обслуговування, даний додаток має високий потенціал для розвитку.

### 1.2 Порівняння з аналогами

З кожним роком, чи навіть днем, Інтернет набирає все більшої популярності. Для деяких це джерело інформації, для інших – місце постійної роботи, ще для когось – інструмент для розваги. Кожен з нас приділяє йому все більшу частину свого часу і проводить все більшу кількість операцій з його допомогою. Популярність Інтернету зростає.

Мобільні додатки для бронювання житла, закладів харчування, транспорту чи навіть побутової техніки є популярними серед людей сьогодні.

Усі звикли здійснювати брнювання через екран смартфона чи комп’ютера в один клік, не витрачаючи час на дзвінки чи зустрічі і не виходячи з домівки.

Серед існуючих реалізацій Android-додатків для бронювання місць у закладах харчування і з можливістю моніторингу якості обслуговування шляхом додавання відгуків, найбільш близькими до створюваного додатку «RestoBooking» є такі:

- TheTable;
- Reservble;
- RestOn;
- Timeat;
- TakeUsEat.

Додаток «TheTable» для бронювання столиків у ресторанах, інтерфейс якого зображено на рисунку 1.1, допомагає користувачам швидко знайти і забронювати столик в ресторані за допомогою свого мобільного пристрою.

До переваг можна віднести швидкість оформлення замовлень та наявність відгуків про заклади, що дає можливість користувачам отримати попереднє враження про заклад і подумати на власний розсуд, чи потрібно відвідувати цей заклад, чи можливо краще обрати інший. До недоліків цього додатку можна віднести довгу процедуру бронювання самого закладу.

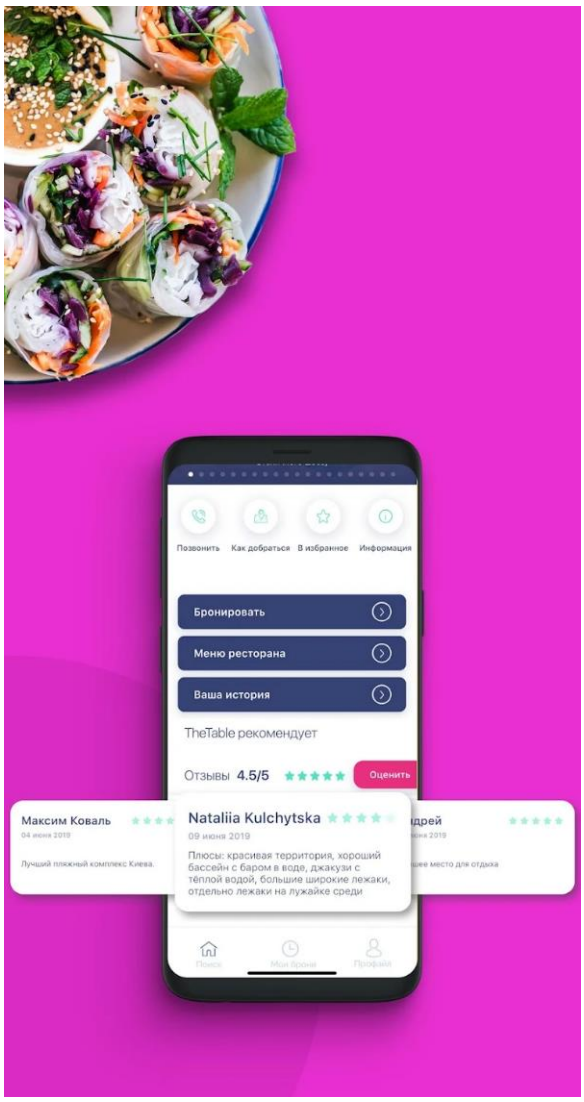



Рисунок 1.1 – Інтерфейс додатку «TheTable»

Для реєстрації закладу необхідно залишити заявку на сайті, після чого з вами зв’яжеться оператор для узгодження всіх деталей. Тобто потрібно витрачати час спочатку на очікування, поки заявка обробиться, а потім дочекатися дзвінка від оператора [6].

«Reservble» – це сервіс, інтерфейс якого зображено на рисунку 1.1, що допомагає обрати заклад на карті (кафе, ресторан, бар тощо) з доступних та забронювати в ньому столик на потрібний час. Додаток немає мобільної версії, тому забронювати столик можливо лише, відвідавши сайт.

Для бронювання столика на сервісі «Reservble» необхідно:

- перейти на сайт «Reservble»;
- обрати потрібне місто;

– зазначити кількість людей та дату;

– на карті вибрати бажаний заклад харчування;

– обрати необхідний час;

– ввести свої дані для бронювання або забронювати через Facebook.

Додаток надає можливість взяти участь у благодійності, для цього необхідно обирати заклади з жовтою позначкою. Одна з основних цілей проєкту – допомога кафе та ресторанам у Києві, Одесі, Харкові та Дніпрі. «Ми об’єднуємо заклади харчування та їхніх гостей для допомоги військовим і постраждалим від російської агресії містам. Бронюючи стіл, ви підтримаєте робочі місця, бізнес, економіку країни та реалізацію гуманітарних та мілітарних цілей проєкту», — описують свою мету Reservble [7].

Також спільно з фондом Сергія Притули та "Життєлюб" сервіс збирає кошти на благодійність, на яку буде йти відсоток від замовлення.

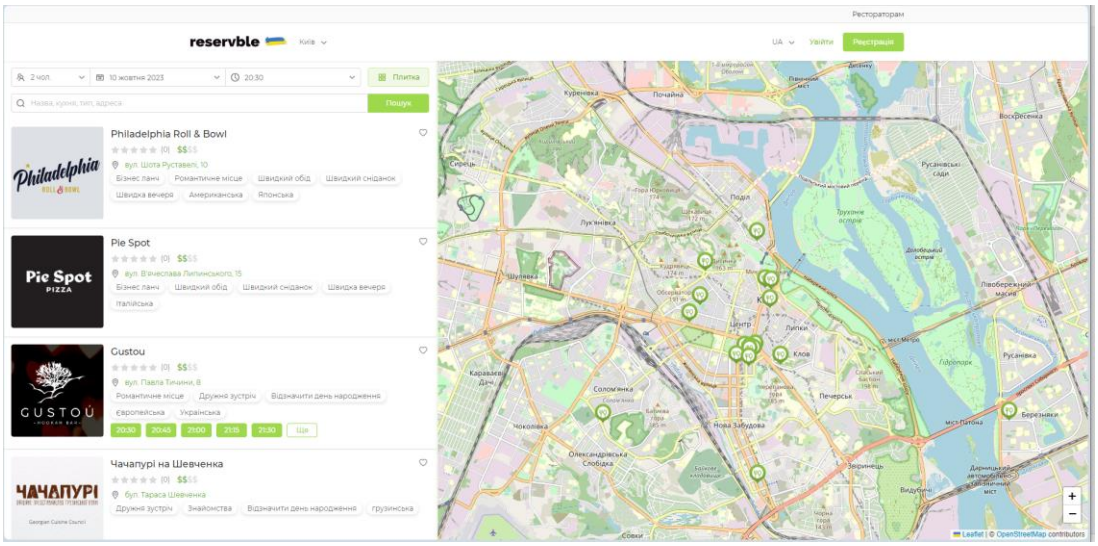



Рисунок 1.2 – Інтерфейс сервісу «Reservble»

До переваг сервісу «Reservble» можна віднести швидкість оформлення замовлень та можливість оплати онлайн. До недоліків цього додатку, можна віднести, знову ж таки, довгу процедуру бронювання самого закладу. Потрібно залишати заявку на сайті, після цього менеджер зв’яжеться з заявником щодо питань реєстрації закладу в системі. Також недоліком сервісу є обмеження

локацій його використання, він підтримує роботу лише в Києві, Львові й Одесі[7].

RestOn – покликаний полегшити і наситити життя українців приємним проведенням часу. Це сервіс, інтерфейс якого зображено на рисунку 1.3, online-бронювання столиків в ресторанах і пабах. З його допомогою користувачі отримують ексклюзивну до 50% знижку на все замовлення, включаючи напої та алкоголь, а заклади, в свою чергу, заповнюють порожні столики, при цьому контролюючи потік відвідувачів. Заклади харчування самі вибирають, коли і в якому розмірі надавати знижки, наприклад, можуть пропонувати гостям 50% знижку вранці в будні або 20-30% знижку на все замовлення у вихідні дні. Крім того, для закладів це відмінний інструмент інформування відвідувачів про свої акції, спеціальні пропозиції, новини, події як постійне джерело донесення необхідної інформації.

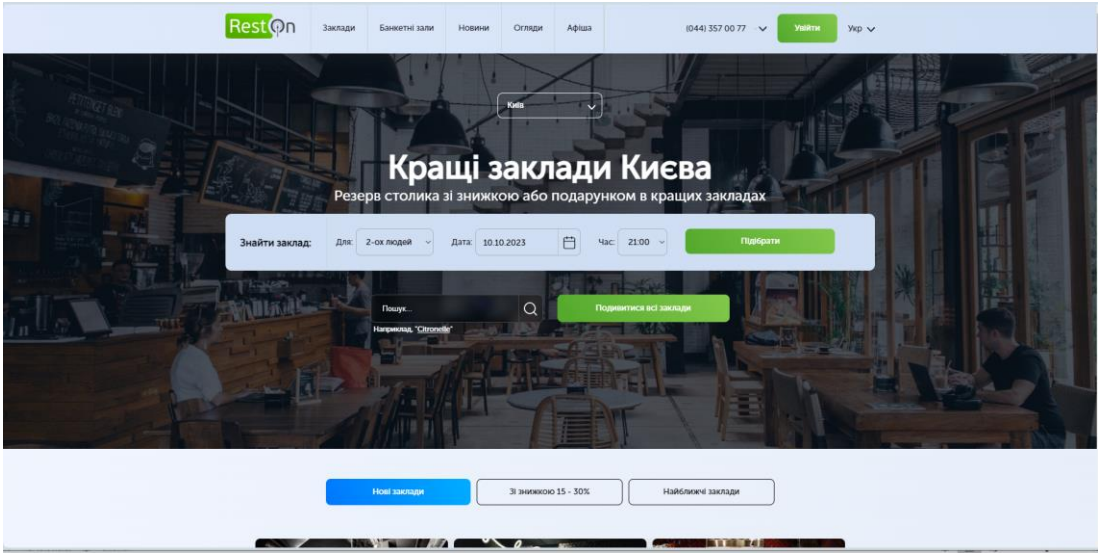



Рисунок 1.3 – Інтерфейс сервісу «RestOn»

Для того, щоб забронювати столик на сайті, потрібно зробити всього лише кілька дій: вибрати заклад, вказати кількість гостей, дату і час відвідин, сплатити комісію на сайті, прийти до закладу в зазначений час і попередити офіціанта про резерв столика через RestOn (повідомити йому код броні). Знижка автоматично враховується при розрахунку. Ніяких купонів, попередніх дзвінків та інших дій

не потрібно. Комісія за резерв одного столика не знімається, кількість осіб за столиком можна вибрати від 1-го до 8-ми. Всі пропозиції можна відфільтрувати за типом кухні, станції метро та середньому чеку, що істотно полегшить пошук відповідного ресторану. Крім того, можна на мапі підібрати найближчий для клієнта заклад.

RestOn.com.ua – це альтернатива купонних сайтів, співпрацювати з якими багато закладів побоюються: потрібно надавати великі знижки, немає можливості контролювати кількість куплених купонів, в результаті чого ресторан не може впоратися з потоком клієнтів.

RestOn.com.ua також надає закладу можливість відключити бронювання на той період, коли заклад і так заповнюється на 100%, або ж, навпаки, включити – коли ресторан пустує через погану погоду чи з інших причин. Для цього заклад повідомляє, коли має вільні столи, або просить прибрати можливість броні і в режимі реального часу може контролювати потік відвідувачів.

Бізнес-модель сервісу RestOn.com.ua побудована за принципом західного аналога www.savored.com, який працює в 10 містах Америки, в його системі більше тисячі ресторанів. На початковому етапі проєкт залучив $750 000 «ангельських» інвестицій, а пізніше ще $3 млн. від Hearst Interactive Media. За 2011 рік кількість ресторанів збільшилася на 414%, кількість користувачів сайту – на 388%, а загальна виручка savored.com – на 551%. У ресторанів-партнерів у сукупності виручка виросла на $25 млн.

RestOn.com.ua – це місце зустрічей закладів та їх відвідувачів. Для закладів харчування – це відмінний шанс заповнити порожні столики у різний час, заявити і більше розповісти про себе на ринку. Для відвідувачів – це можливість вибрати своє улюблене місце або розвідати для себе новий заклад та забронювати столик у ньому, а на додаток отримати до 50% знижку. Основним недоліком цього продукту є відсутність мобільної версії, що значно зменшує зручність користування сервісом [8].

«Timeat» — сервіс попереднього замовлення страв через мобільний додаток (див. рисунок 1.4). Користувач може замовити страви й напої на вказані

час і кількість людей в обраному закладі. За ним бронюється стіл, у вказаний час їжа вже готова й клієнт не чекає, коли приходить до закладу.

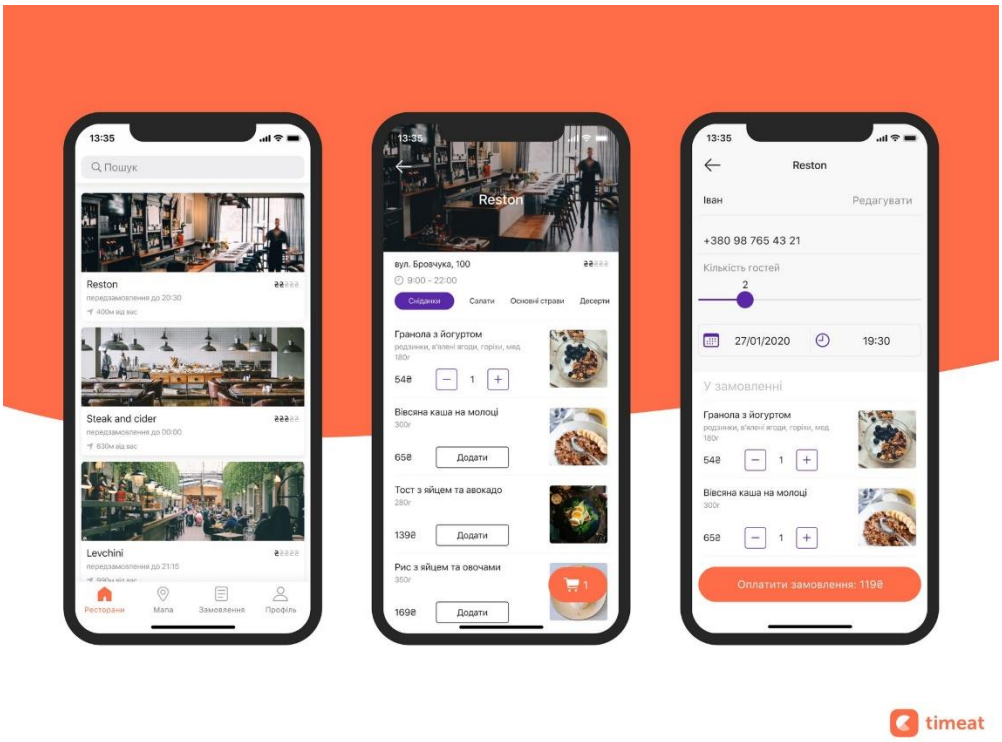



Рисунок 1.4 – Інтерфейс сервісу «Timeat»

Як працює сервіс:

– користувач оформлює та оплачує замовлення через програму;
– ресторан отримує замовлення та бронює стіл;
– клієнт приходить на вказаний час та отримує готове замовлення.

За словами команди timeat, зі статистикою з ресторанів в Україні є великі проблеми. Наприклад, оборот української ресторанної індустрії становить приблизно $1,4 млрд, а згідно з дослідженням 2018 року лише 25% домогосподарств в Україні можуть дозволити собі їсти не дома.

Першими ідею попереднього замовлення та миттєвого бронювання реалізувала команда Allset у 2014. Місцевий продукт називався Settle. Через два роки розробники прибрали його з доступу та запустили сервіс лише для американського ринку, де він успішно розвивається. У 2019 році запускався проєкт RestYou. З його допомогою теж можна було бронювати стіл та замовляти

страви. Однак він припинив роботу, пославшись на маленький ринок. Команда timeat з цим не згодна [9].

«TakeUsEat» – це сервіс онлайн-бронювання (див. рисунок 1.5). З TakeUsEat можна миттєво бронювати столики в улюблених ресторанах без телефонних дзвінків. Є можливість знайти місця, які будуть до душі, використовуючи розумний пошук.

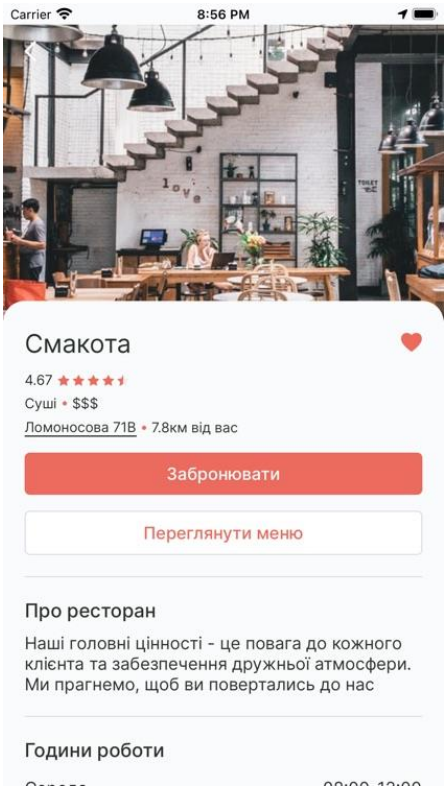



Рисунок 1.5 – Інтерфейс сервісу «TakeUsEat»

Вже налаштована система проста у використанні та повністю адаптована під потреби ресторанів: історія відвідувань, телефонних розмов та нотаток. Зручна візуалізація залів з переліком бронювань завжди під рукою. Знайти вільне місце та відслідкувати заповненість залу стало набагато простіше. Персональна сторінка закладу з меню та можливістю залишити відгук є доступною для відвідувачів. Також є можливість керувати одночасно декількома закладами та переспрямовувати гостей між закладами мережі. Зручна сітка замість заплутаної таблиці у блокноті, що дає можливість редагувати тривалість експлуатації та столики бронювань без додаткових кліків. Але цей додаток є платним для

рестораторів, тобто для того, щоб зареєструвати заклад у системі, необхідно заплатити кошти[10].

Після аналізу усіх аналогів визначено їхні переваги й недоліки та проведено порівняння з власним Android-додатком, який розробляється. Результат порівняння зведено в таблицю 1.1.

Таблиця 1.1 – Порівняльні характеристики мобільних додатків

| Назва додатку / Критерій | TheTable | Reservble | RestOn | Timeat | TakeUsEat | Resto Booking |
|---|---|---|---|---|---|---|
| Наявність мобільного додатку | + | - | - | + | + | + |
| Пошук закладу за тегами під мету відвідування | - | + | - | - | - | + |
| Швидка реєстрація закладу без дзвінка менеджера | - | - | - | - | + | + |
| Безкоштовність | + | + | + | + | - | + |
| Можливість додавання закладу у будь-якій локації | + | - | - | - | + | + |
| Підсумковий результат | 3 | 2 | 1 | 2 | 3 | 5 |

Отже, можна зробити висновок, що у порівнянні з існуючими аналогами,

розробка власного мобільного додатку є доцільною. В результаті роботи отримаємо додаток, який буде покривати недоліки існуючих на даний момент аналогів та забезпечить вищу ефективність роботи.

### 1.3 Аналіз методів розв'язання задачі

У минулому в різних частинах світу були золоті лихоманки. Тепер очевидно, що сучасна епоха переповнена одержимістю мобільними пристроями. Широке використання мобільних телефонів зростає з неймовірною швидкістю, і компанії, які їх виробляють, інвестують великі суми грошей у просування та розвиток мобільних технологій.

Розробка додатків для смартфонів є швидкозростаючим сектором програмування, оскільки кількість мобільних телефонів значно перевищує кількість комп'ютерів.

Процес створення програм спеціально для певної платформи та пристрою широко відомий як нативна розробка, або ж розробка кросплатформних чи гібридних програм з використанням таких фреймворків, як ReactNative, Flutter і Xamarin [11].

Під поняттям «нативна розробка» розуміється розробка програми для смартфона на конкретній мові програмування під конкретну операційну систему. Такі додатки є доволі продуктивними і не мають обмежень у розробці. Для нативної розробки в ОС Android використовується Java або Kotlin, тоді як iOS покладається на Swift.  Серед переваг такого підходу до розвитку є створення впізнаваного інтерфейсу користувача для конкретної платформи [12].

Однак недоліки використання цього підходу можна окреслити як витрати на розробку та підтримку програм, які часто є досить значними, процес написання програми потребує більше часу. Розробка програми для різних операційних систем – це трудомісткий процес. Він не тільки дорогий. Процес розробки та тестування може бути тривалим через унікальні вимоги до кожного продукту. Процес розробки програм, які можуть працювати на кількох платформах, стає можливим завдяки використанню таких веб-технологій, як

HTML, CSS і JavaScript. Ці технології дозволяють створювати додаток, який може працювати на різних платформах одночасно. Одна з переваг цього методу розробки полягає в тому, що він економічно ефективний, часто для виконання завдання потрібен лише один спеціаліст. Однак є також недоліки, які слід враховувати, такі як можливі затримки у відповіді на введення користувача та зниження загальної продуктивності роботи програми.

Створення приємного та зручного інтерфейсу користувача великою мірою визначає успіх того чи іншого додатку. Jetpack Compose – технологія, яка дозволяє будувати сучасні інтерфейси швидко і ефективно, набуває все більше популярності серед Android-розробників.

На відміну від XML, свого попередника, Compose закладає нову філософію створення інтерфейсу користувача і його відповідного застосування у додатках Android [13]. Так розробники описують необхідні властивості у composable-функції та визначають state (поточний стан) елементів, на основі чого система створює та відображає той чи інший інтерфейс на екрані [14]. Елементи додатків у Compose є функціями, а івенти змінюють стани і відповідно сам інтерфейс. Важливим компонентом Compose є Slot API із резервом місця у контейнері, у якому можна розмістити текст, кнопку, іконки або малюнки без обмежень для розробника. Compose застосовує макети, які за розташуванням схожі до макетів XML, зокрема, Box, Row, Column, LazyColumn (див. рисунок 1.6).

Окрім цього, у Compose використовується макет ConstraintLayout, який надає можливість задавати параметри елементам, а також розташовувати і відображати їх на екрані у бажаному порядку [15]. Макет Scaffold у свою чергу забезпечує розміщення усіх елементів на екрані у звичному місці у Android-додатках [16].

Параметр composable-функції, modifier, описує багато параметрів елементів інтерфейсу, які потрібно відобразити розробнику. Існує багато методів його використання з іншими  елементами інтерфейсу. При цьому у разі вказівки невідповідних властивостей до елементів інтерфейсу modifier не дозволить це зробити, якщо певний елемент не підтримує ту чи іншу властивість [17].

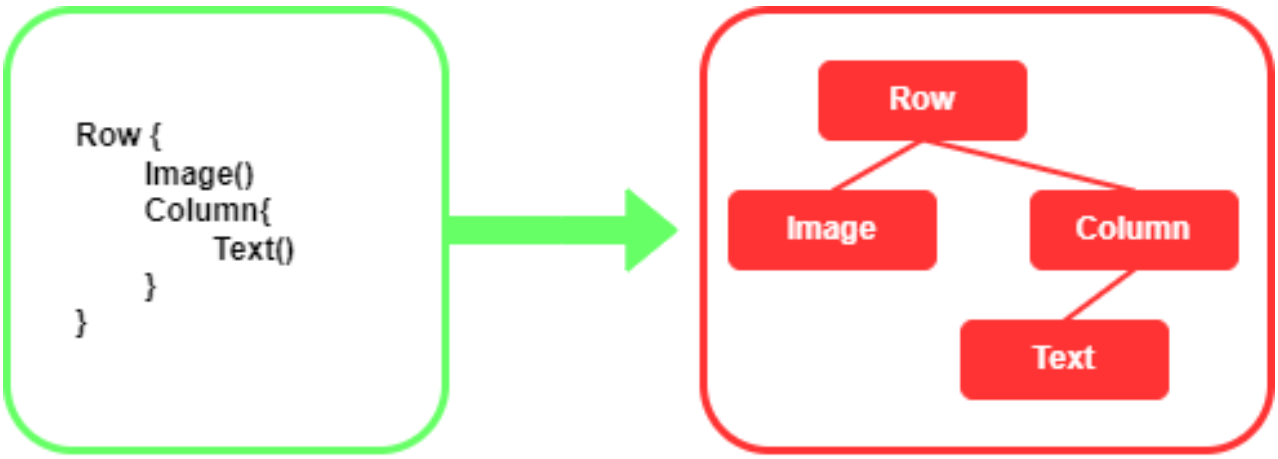



Рисунок 1.6 – UI дерево інтерфейсу користувача

Оскільки Android виконує функції поступово і в певному порядку, важливо задавати параметри чітко і уважно. У кожній composable-функції першим параметром бажано задавати modifier для визначення властивостей, які мають містити певні кнопки чи іконки. Таким чином, ці modifiers можуть бути застосовані до всіх внутрішніх функцій, які будуть використовуватися і надалі.

Додатковою важливою особливістю Compose є функція Preview, яка застосовується і в XML. Хорошою практикою вважається розбивати функції на якомога менші одиниці під час створення інтерфейсу за допомогою Compose для того, щоб той чи інший елемент міг відображатися окремо. Після цього можна задати ще одну composable-функцію, написати до неї Preview-анотацію, а Compose у свою чергу відобразить задані елементи на екрані. При цьому варто зазначити, що під час зміни окремих параметрів можна буде побачити, як буде виглядати інтерфейс користувача, що створюється [18].

Зі станів у Compose використовується MutableState (див. рисунок 1.7), який задає дані, на які буде спиратися інтерфейс і відображатися відповідним чином на екрані [19]. Стани можна створювати всередині певної composable-функції та передавати за допомогою параметру. У разі зміни стану, composable-функція зміниться відповідно до нових даних. Таким чином, під час рекомпозиції інтерфейсу будуть викликатися лише ті функції, у яких змінився якийсь стан. Дані, які знаходяться в інших моделях, наприклад, у XML, можна легко передати у функції на Compose, оскільки він підтримує різні способи використання та

збереження даних (Rx2, flow та інші) за допомогою провайдерів. Компоновані функції можуть використовувати remember API для збереження об'єкта в пам'яті. Значення, обчислене методом remember, зберігається в композиції під час початкової композиції, а збережене значення повертається під час рекомпозиції. Remember можна використовувати для зберігання як змінних, так і незмінних об'єктів. Remember зберігає об'єкти в композиції та забуває об'єкт, коли складовий, який викликав запам'ятовування, видаляється з композиції [20].

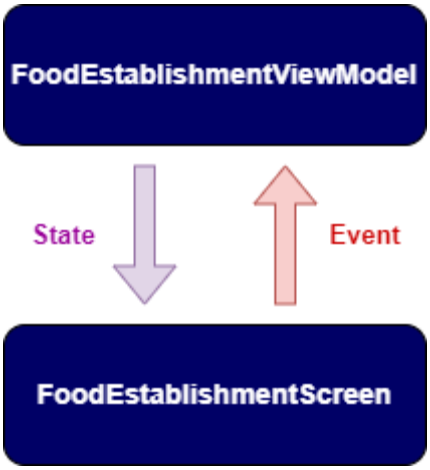



Рисунок 1.7 – Схема роботи state у додатку

Використання Compose для створення інтерфейсу користувача є дуже ефективним та менш енергозатратним. Так, під час побудови інтерфейсу за допомогою XML потрібно створити клас Kotlin, а також XML-файл та поєднати їх. Після цього потрібно вказати елементи інтерфейсу, описати властивості та поєднати ці елементи зі View- моделями та іншими ресурсами, а також задати характеристики для бажаного відображення. Застосовуючи Compose, розробнику потрібно лише створити клас Kotlin, описати функції з відображенням і передати туди всі дані із характеристиками. Compose значно полегшує та пришвидшує цей процес, оскільки з ним зникає потреба створювати XML-файли, які зазвичай є достатньо великими. Compose надає можливість застосовувати усі функції Kotlin, при цьому використовуючи менше коду. Режим Preview у Compose є дуже зручним, так як на будь-яку довільну composable-функцію можна написати Preview-функцію і одразу подивитися, як вона

виглядає. Також Jetpack Compose дозволяє комбінувати різні composable-функції на різних макетах із різними параметрами, що неможливо зробити в XML.

Однак, варто зазначити, що перехід на Compose може бути непростим у разі, якщо міграцію потрібно провести у рамках великого проєкту. Це передбачає тривалу роботу із детальним і уважним описом окремих елементів інтерфейсу. Оскільки Compose ще досі знаходиться у процесі розробки, оновлення та вдосконалення, на даний момент він ще не підтримує усі свої функції. Незважаючи на це, не прогнозується наявність проблем з його використанням.

Популярність Compose буде лише зростати, оскільки він має безліч переваг. Великі відомі компанії, такі як Airbnb, Booking, Lyft, Twitter та інші, уже застосовують Compose, а менші бізнеси поступово будуть переходити на цю технологію. Майже всі Android-розробники починають створювати нові рішення на Compose за умови відсутності чітких обмежень з боку замовника. Офіційні джерела рекомендують детальніше вивчати Compose, оскільки він буде інтегрований у нові проєкти, а його знання у майбутньому стане однією з вимог вакансій у IT-сфері.

### 1.4 Постановка задач дослідження

Провівши аналіз стану розвитку Android-систем бронювання місць і моніторингу якості обслуговування в закладах харчування «RestoBooking», було визначено задачі, які необхідно вирішити у процесі розробки програмного продукту:

- розробити методи і модель роботи програми;
- реалізувати реєстрацію та авторизацію користувачів;
- розробити метод додавання нового закладу;
- розробити метод пошуку закладів харчування;
- розробити метод моніторингу якості обслуговування;
- розробити метод бронювання місць у закладах харчування;
- розробити Android-систему бронювання місць і моніторингу якості обслуговування в закладах харчування «RestoBooking»;

- розробити зручний та зрозумілий графічний інтерфейс;
- провести тестування Android-системи.

### 1.5 Висновки

У першому розділі було розглянуто стан мобільних систем бронювання місць і моніторингу якості обслуговування в закладах харчування. Було проведено порівняння таких аналогів як TheTable, Reservble, RestOn, Timeat та TakeUsEat.. Результати аналізу вказують на недостатність наявного функціоналу для пошуку закладів харчування і бронювання місць, а також для опису і реєстрації закладів харчування. Було проаналізовано способи розробки мобільних додатків, у результаті аналізу та порівняння переваг і недоліків було обрано метод «нативної розробки». Також було проаналізовано способи написання графічного інтерфейсу XML і Jetpack Compose, в результаті аналізу та порівнянь переваг та недоліків було обрано Jetpack Compose. Було сформульовано основні задачі, які потрібно вирішити протягом розробки мобільного додатку під платформу Android.

# 2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ СИСТЕМИ

## 2.1. Архітектурне проєктування програмної системи

При розробці мобільного Android-додатку було використано один з найпопулярніших шаблонів програмування Model-View-ViewModel, або скорочено MVVM [21]. Цей шаблон програмування використовується для розробки користувацького інтерфейсу і розділяє між собою бізнес логіку та логіку відображення. Схема взаємодії Model, View і ViewModel у мобільному додатку «RestoBooking» зображено на рисунку 2.1.

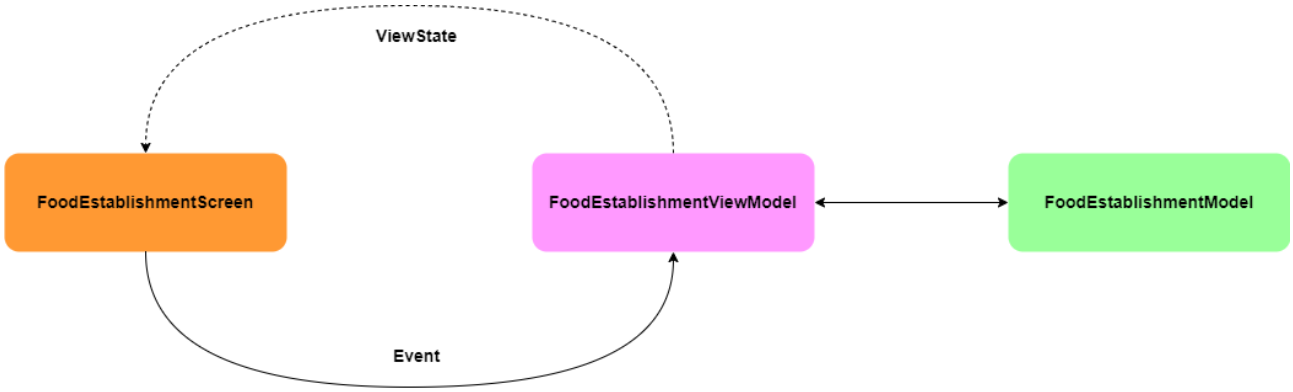



Рисунок 2.1 – Схема взаємодії Model, View і ViewModel у шаблоні MVVM

MVVM – це шаблон архітектури, який розділяє код на три частини: модель, відсік і ViewModel. Модель відповідає за зберігання даних і їх логіку. Наприклад, Model може зберігати список товарів або список користувачів View відповідає за відображення даних користувачеві. Наприклад, View може відображати список товарів у вигляді списку або таблиці. ViewModel відповідає за зв'язок між Model та View. ViewModel отримує дані від Model і передає їх View.

MVVM має низку переваг, включаючи:

– код розділений на три основні частини, що полегшує розуміння та підтримку додатка;

– кожен компонент можна тестувати окремо, що полегшує виявлення помилок;

− Android Studio має вбудовану підтримку MVVM, що полегшує реалізацію шаблону.

MVVM також має деякі недоліки, включаючи:

− MVVM вимагає додаткового коду для реалізації шаблону;

− MVVM може бути складним для розуміння та реалізації.

MVVM – це популярний шаблон архітектури для розробки Android-додатків. Він пропонує низку переваг, включаючи розділення відповідальності, тестування та підтримку MVVM. Однак MVVM також вимагає додаткового коду та може бути складним для розуміння та реалізації.

Для реалізації графічного інтерфейсу було обрано Jetpack Compose. Jetpack Compose – це набір інструментів для побудови сучасних UI в Android-додатках. Компанія Google анонсувала Jetpack Compose в 2019 році, а вже в березні 2021 з’явилася бета-версія фреймворку [22].

З використанням Jetpack Compose стало простіше працювати над UI для Android. Прості API допомагають створювати приємні для ока інтерфейси. Кількість коду зменшується загалом на 30%. Швидкість розробки теж скорочується, відбувається оптимізація UI завдяки потужним функціям: Composable Preview, Editor Actions, Layout inspector, Iterative code development, Animations.

Jetpack Compose побудований на базі мови програмування Kotlin від компанії JetBrains. Kotlin особливо популярний у мобільній розробці та дає вагомі переваги для Compose. Наприклад, є можливість написання data class для використання в стейтах, дефолт-значення – методи, extension functions – щоб розширювати функціональність [23].

В основі технології Jetpack Compose є декларативний підхід, тобто відбувається без xml-layouts. Цей факт характеризує декларативний UI. Тобто замість перерахування послідовності дій прямим текстом описується, яким повинен бути елемент інтерфейсу. Після цього платформа виконує дію з урахуванням контексту та дефолтних значень.

Jetpack Compose існує на базі робочого середовища Android Studio [24]. На

платформі реалізовано функціонал спеціально для бібліотеки Compose. Android Studio пропонує багато нових функцій персонально для Jetpack Compose. Вона використовує підхід code-first. Це дозволяє підвищити продуктивність розробника, не вимагаючи вибирати між редактором коду та інтерфейсом дизайну.

Ключова різниця між інтерфейсом користувача на базі Android Views і Jetpack Compose полягає в тому, що останній не спирається, власне, на views для рендерингу композитних елементів. Завдяки такому архітектурному підходу, Android Studio дає розширені можливості для Jetpack Compose без необхідності відкривати емулятор або підключатися до пристрою.

Переваги Jetpack Compose:

− Декларативний UI. Jetpack Compose використовує декларативний синтаксис, який легко читати та розуміти. Це безпосередньо впливає на швидкість та якість розробки;

− Простий в освоєнні. Фреймворк має низький поріг входження, тому що спрощує розробку інтерфейсів і допомагає скоротити обсяг коду;

− Тестування UI. Jetpack Compose полегшує написання тестів інтерфейсу користувача на Kotlin. За рахунок цього життя програміста стає простішим. Він може переконатися, що інтерфейс працює коректно та без помилок;

− Інтерактивний UI. Jetpack Compose забезпечує оновлення в реальному часі при взаємодії користувача з інтерфейсом. Таким чином, додаток виглядає більш привабливо в очах користувача і його досвід користувача покращується;

− Повторне використання. За допомогою Jetpack Compose можна створювати багаторазово використовувані компоненти UI, а потім використовувати їх на кількох екранах і навіть у різних програмах. Це особливо важливо при створенні складних інтерфейсів користувача.

− Гнучкість. Jetpack Compose пропонує багато можливостей для проектування UI. Програмісти можуть легко налаштовувати макети, додавати анімації та кастомізувати зовнішній вигляд програми, не

вдаючись до написання великої кількості коду.

Недоліки Jetpack Compose:

− Підтримує лише Kotlin. Це може бути недоліком для розробників, які вже впевнено володіють іншими мовами програмування, наприклад Java, C++ або Swift. Вони можуть не захотіти вкладати час та сили у вивчення нової мови. Це також може бути проблемою для команд розробників із різними мовними уподобаннями або рівнями навичок. Крім того, не всі сторонні бібліотеки чи інструменти, які розробники забажають використати у своїх проектах, можуть бути написано на Kotlin. Тому виникають проблеми сумісності або додаткова робота, необхідна їх інтеграції з Jetpack Compose;

− Проблеми з відображенням UI-компонентів. Під час рендерингу можуть виникати лаги та затримки. Це пов'язано з тим, що Compose призначений для динамічного рендерингу уявлень, на відміну від традиційного підходу з використанням попередньо намальованих XML-макетів. Це означає, що процес рендерингу може тривати більше часу, особливо під час роботи з анімацією. Однак проблеми з продуктивністю часто можна пом'якшити, оптимізувавши розташування UI-компонентів та мінімізувавши кількість рекомпозицій, що відбуваються в процесі рендерингу.

− якщо у вас є готова програма на XML-макетах, то перехід на Jetpack Compose може вимагати великих змін коду, а це триватиме довго.

Для реалізації серверної частини було обрано Firebase. Firebase – це платформа розробки мобільних та веб-додатків, яка надає повний спектр інструментів для збереження та обробки даних в хмарному середовищі.

Firebase надає можливість швидко розгорнути сервер, забезпечити безпеку даних та ефективно взаємодіяти з базою даних. Таким чином, вибір Firebase як серверного рішення є стратегічним кроком у розвитку програмного застосунку.

### 2.2 Розробка методу реєстрації нового закладу

Реєстрація нового закладу є невід'ємною частиною мобільного додатку «RestoBooking». Для того, щоб розпочати реєстрацію нового закладу в системі, необхідно перейти у вкладку «Профіль», натиснувши на відповідну іконку в нижній частині екрану, як зображено на рисунку 2.2. Наступним кроком відкриється перше вікно реєстрації закладу, як продемонстровано на рисунку2.2.

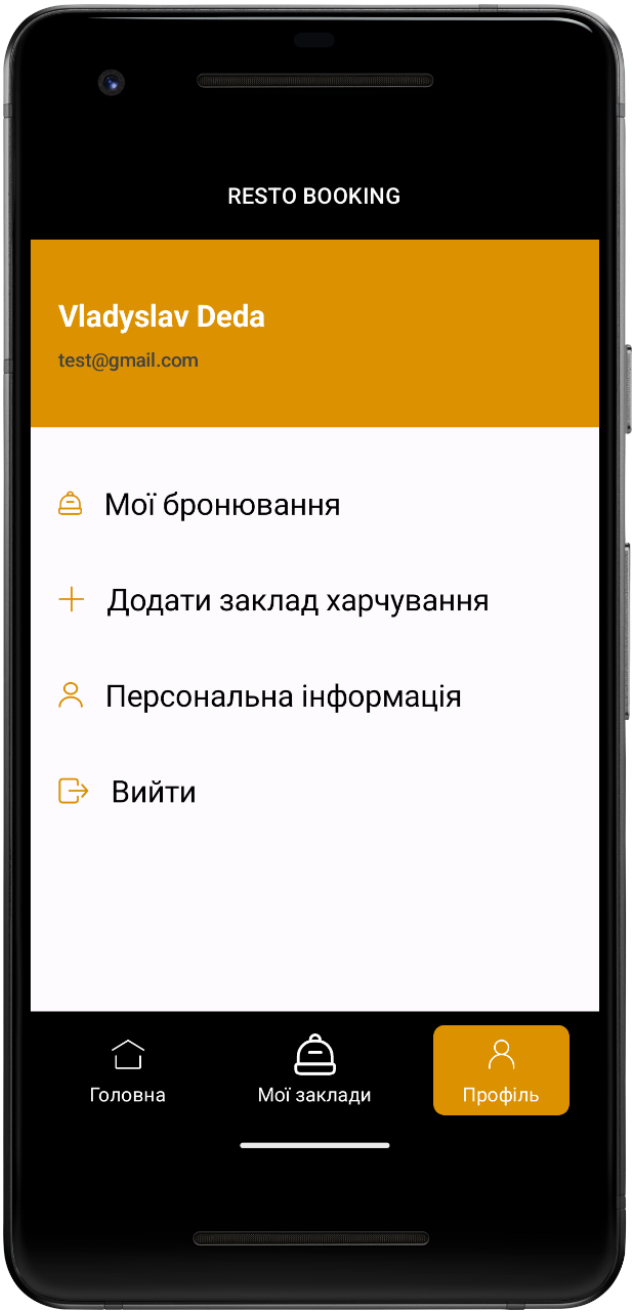



Рисунок 2.2 – Екран «Профіль»

Метод додавання нового закладу харчування в систему передбачає послідовність виконання функцій:

1. Натискається кнопка «Додати новий заклад», OnClickListener якої відкриває новий екран через Navigation Graph.
2. Заповнюються поля у TextField «Назва закладу», «Місто», «Адреса закладу», «Телефон для бронювання», Опишіть заклад» які зберігаються у MutableStateFlow.
3. Із DropDownMenu обрається з меню «Тип закладу».
4. Через MaterialTimePicker обирається «Початок роботи».
5. Через MaterialTimePicker обирається «Кінець роботи».
6. Натискається кнопка «Продовжити», яка стає активною коли при оновлені State всі поля заповнені.
7. Обираються теги закладу, View яких має визначений розмір і має можливість прокрутки з-за допомогою rememberScrollState().
8. Натискається кнопка «Продовжити», яка стає активною, коли у uiState визначено, що вибрано хоча б один тег.
9. Обираються фотографії закладу (не більше 6, не менше 1), відкриваючи галерею і використовуючи rememberLauncherForActivityResult, у якого contract = ActivityResultContracts.PickVisualMedia().
10. Натискається кнопка «Зареєструвати», яка стає активною, коли обрана хоча б одна фотографія.
11. Фотографії по черзі завантажуються на сервер, формуючись у об’єкти document.
12. Отримується downloadUrl кожної фотографії, за якою в майбутньому завантажуються фотографії.
13. Заклад добавляється з доданою інформацією і списком downloadUrl фотографій.
14. Перехід на екран «Профіль».

Блок-схему алгоритму розглянутого метода подано на рисунку 2.3.

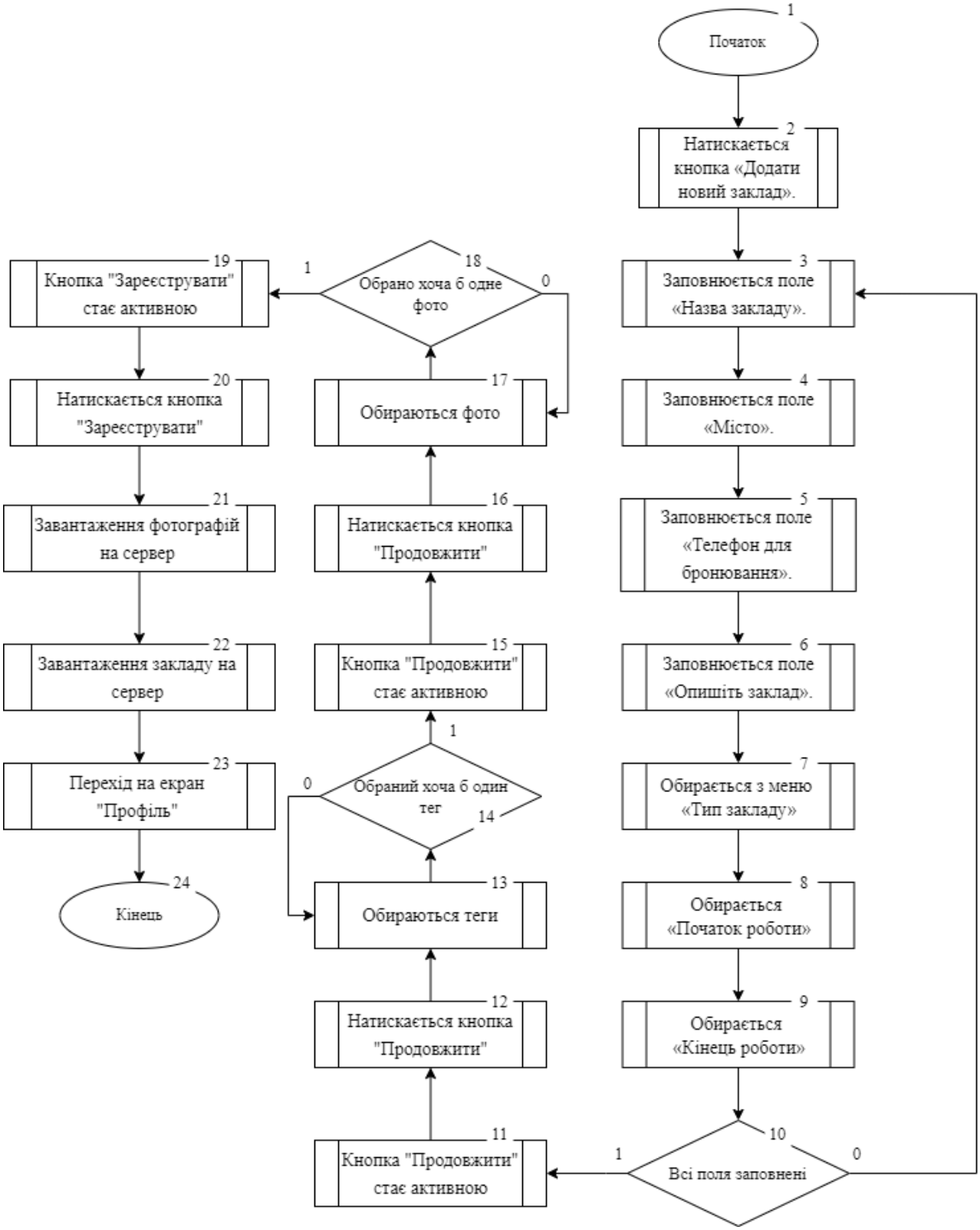



Рисунок 2.3 – Блок-схема алгоритму додавання нового закладу харчування в систему

Код реалізації додавання нового закладу в систему подано на рисунку 2.4.

```
override suspend fun registerFoodEstablishment(foodEstablishment: FoodEstablishment): Result<Unit> =
    withContext(Dispatchers.IO) { this: CoroutineScope
        try {
            val imagesUrls = mutableListOf<String>()
            foodEstablishment.photoList.forEach { it: Photo
                val ref = storage.child( pathString: "${foodEstablishment.name}_${it.index}")
                it.uri?.let { it1 -> ref.putFile(it1).await() }
                val url: String = ref.downloadUrl.await().toString()
                imagesUrls.add(url)
            }
            firestore.collection(FOOD_ESTABLISHMENT_COLLECTION)
                .add(foodEstablishment.toDto(imagesUrls)).await()
            Result.success(Unit) ^withContext
        } catch (e: Exception) {
            Result.failure(e) ^withContext
        }
    }
```

Рисунок 2.4 – Код реалізації додавання нового закладу харчування в систему

На рисунку видно, що на вхід функція отримує об’єкт класу FoodEstablishment (див. рисунок 2.5).

```
data class FoodEstablishment(
    val name: String = "",
    val city: String = "",
    val address: String = "",
    val phoneForBooking: String = "",
    val description: String = "",
    val foodEstablishmentType: FoodEstablishmentType = FoodEstablishmentType.Default,
    val tags: List<String> = emptyList(),
    val selectedTimeFrom: Date? = null,
    val selectedTimeTo: Date? = null,
    val photoList: List<Photo> = emptyList(),
    val ownerName: String = "",
    val rating: Int = 0,
    val comments: List<String> = emptyList()
)
```

Рисунок 2.5 – Код класу FoodEstablishment

Далі фото додається на сервер і після цього отримується downloadUri кожного фото, з-за допомогою якого потім будуть отримуватись завантажені фотографії. Потім downloadUri добавляються до об’єкта FoodEstablishment і завантажуються на сервер з колекцію food_establishments.

### 2.3 Розробка методу пошуку закладів харчування

Функція пошуку закладів харчування є однією з базових функцій, які повинні бути реалізовані у мобільному додатку «RestoBooking». Екран пошуку закладів відкривається першим після авторизації чи реєстрації користувача. На

рисунку 2.6 зображено головний екран з можливістю пошуку закладів харчування.

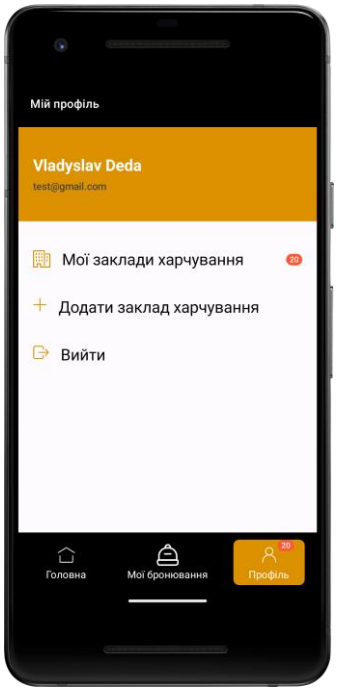



Рисунок 2.6 – Головний екран з можливістю пошуку закладів харчування

Метод пошуку закладів харчування акумулює функціонал:

1. Відкривається головний екран програми.
2. Перевіряється чи є збережені критерії пошуку з попереднього пошуку.
3. Показується діалогове вікно на якому користувач може застосувати попередні критерії, або ввести нові.
4. Завантажуються теги, які доступні у системі, які перетворюються у Set не повторюваних значень.
5. У TextField вводиться місто, у якому користувач хотів би обрати заклад, яке зберігається у MutableStateFlow.
6. Обираються теги зі списку доступних, при кліку на які спрацьовує OnClickListener, і дані у uiState про обрані теги оновлюються.
7. Заповнюються час і дата замовлення через MaterialDatePicker і MaterialTimePicker.
8. Якщо всі поля заповнено, оновлюється MutableStateFlow і значення

кнопки «Пошук» enabled стає true.

9. Натискається кнопка «Пошук».

10. Відкривається екран списку закладів через Navigation Graph.

11. Відповідно до обраних критеріїв в персоналізованому режимі для кожного користувача робиться запит GET до бази Firebase для отримання списку закладів.

12. Заклади з найвищими рейтингами будуть відображатись вище у списку.

Блок-схема алгоритму пошуку закладів відповідно до критеріїв, які обрав користувач, зображено на рисунку 2.7.

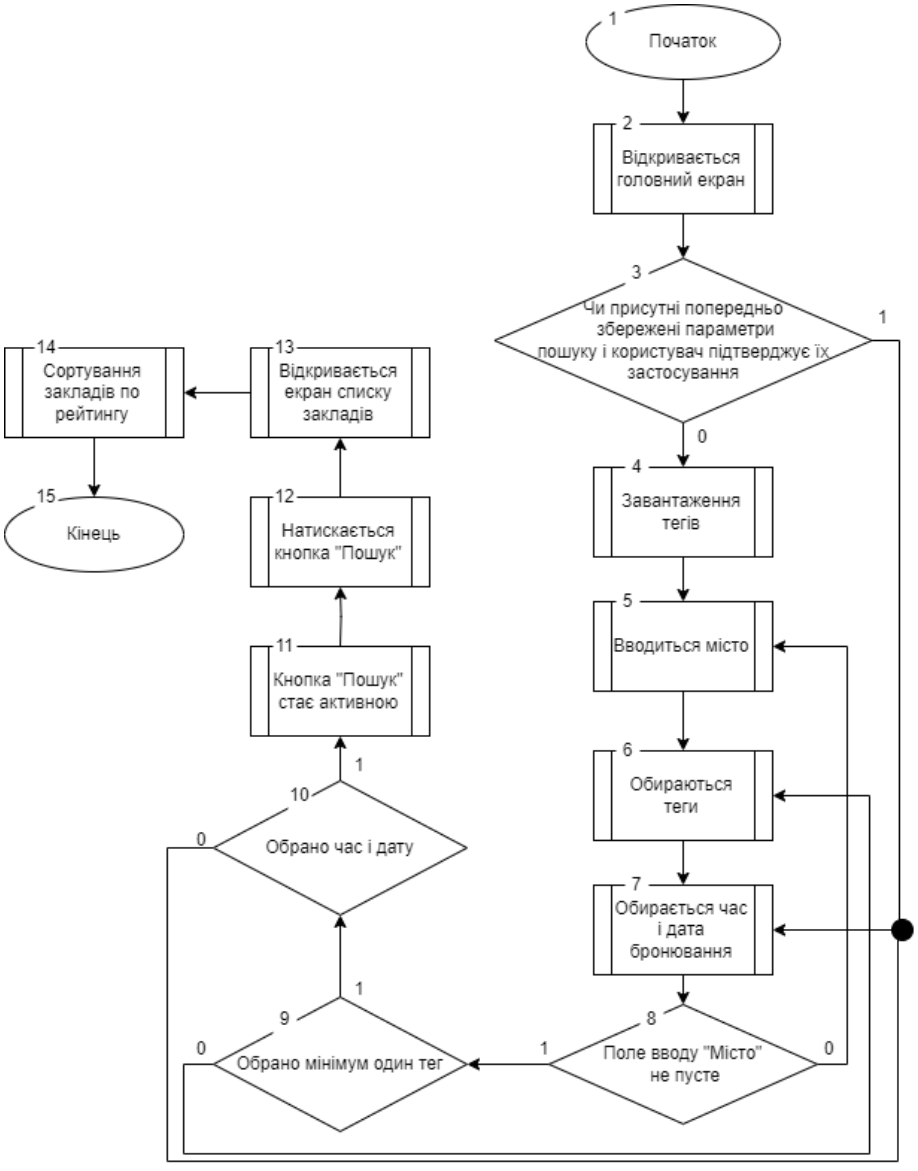



Рисунок 2.7 – Блок-схема алгоритму пошуку закладів

Після завантаження закладів харчування, вони відображаються у списку, як продемонстровано на рисунку 2.8.

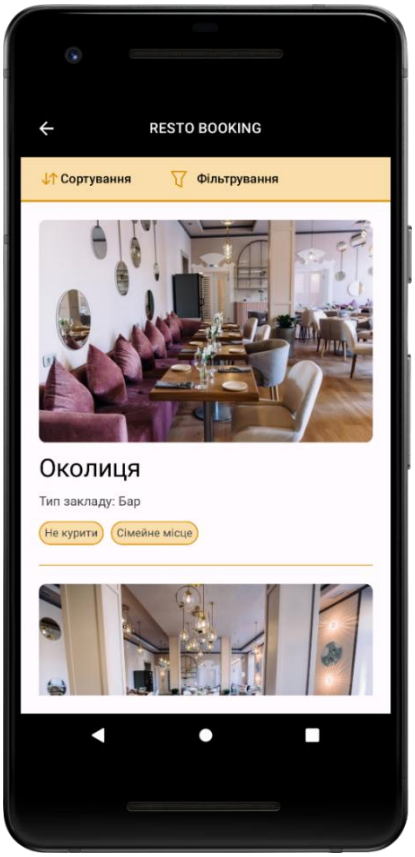



Рисунок 2.8 – Екран списку закладів

На цьому екрані користувач може побачити основну інформацію про заклад, а саме: головне фото, назву, тип закладу, теги і, якщо є відгуки про якість обслуговування, на основі яких формується рейтинг, то буде відображено рейтинг. Для перегляду більш детальної інформації про заклад, можливості додати відгук, або ж забронювати місце, потрібно натиснути на нього у загальному списку закладів.

### 2.4 Розробка методу моніторингу якості обслуговування

Для реалізації моніторингу якості обслуговування у закладах харчування було вирішено розробити функціонал для додавання відгуків і рейтингів про заклад. Діалогове вікно, з допомогою якого можна залишити оцінку про

обслуговування в даному закладі, а також текстовий відгук, зображено на рисунку 2.9.

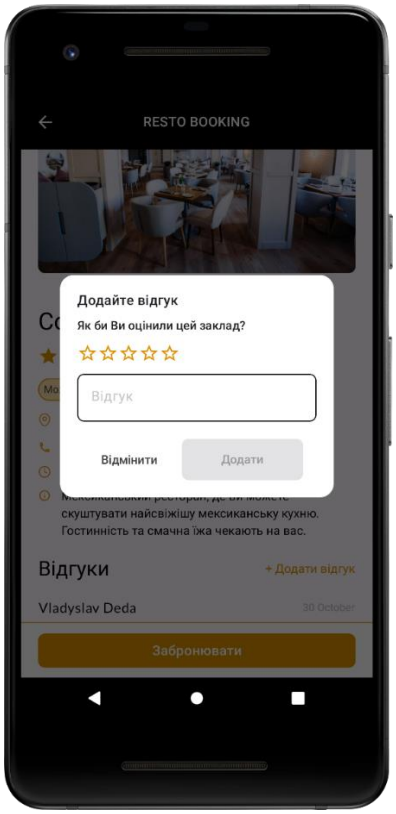



Рисунок 2.9 – Діалогове вікно додавання відгуків

Здебільшого користувачі не полюбляють витрачати багато часу на те, щоб залишити відгук про заклад, товар чи якість обслуговування, тому просто пропускають цей крок, якщо є така можливість, або й зовсім закривають додаток. Для того, щоб уникнути дискомфорту для користувачів, обов'язковим є лише поле рейтингу, після вибору якого, кнопка «Додати» стає активною. Тобто, введення поля «Відгук» є необов'язковим і оцінити якість обслуговування можна і без текстового відгука.

Після того, як коментар буде додано і діалогове вікно, в якому додано коментар, буде закрито. Далі, відкриється наступне діалогове вікно з опитуванням для статистики закладу і моніторингу якості обслуговування у ньому. Для того, щоб заповнення відповідей не займало багато часу, потрібно просто обрати RadioButton із групи запропонованих відповідей. Діалогове вікно

опитування зображено на рисунку 2.10.

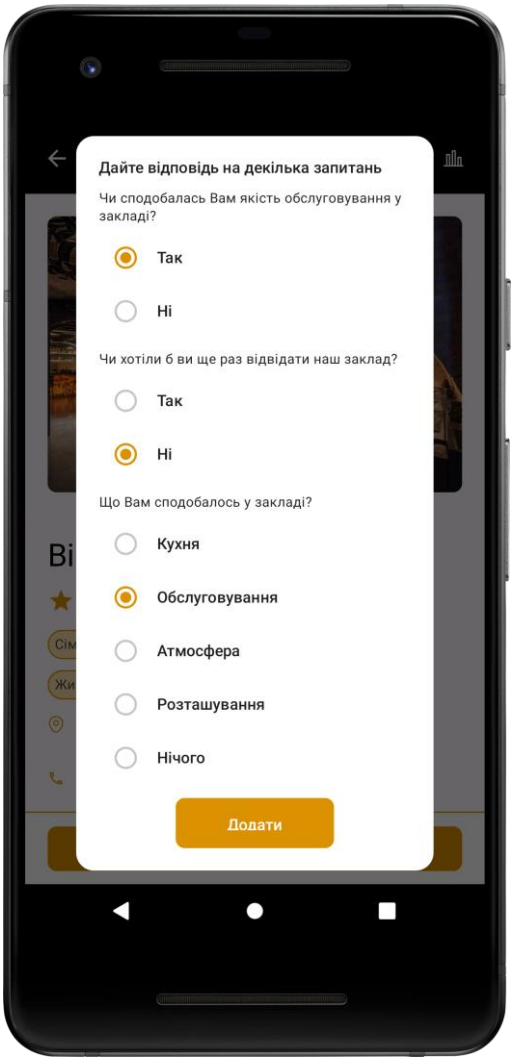



Рисунок 2.10 – Діалогове вікно додавання статистики

Метод оцінювання якості обслуговування включає послідовність дій:

1. Натискається кнопка «Додати відгук», clickListener якої, в свою чергу, відкриває діалогове вікно додавання відгуків.
2. На RatingBar виставляється рейтинг від 1 до 5.
3. У полі TextField з inputType Text пишеться відгук.
4. Статус enabled у кнопки стає true.
5. Натискається кнопка «Додати».
6. Відгук надсилається на сервер методом POST.
7. Відкривається діалогове вікно опитування для статистики про заклад.

8. Обираються RadioButton для вибору відповіді на питання.

9. MutableStateFlow оновлюється і параметр кнопки «Додати» стає true.

10. Опитування надсилається на Firebase і додається до статистики закладу.

11. Оновлюється інформація про заклад і доданий відгук відображається у загальному списку усіх відгуків.

12. Оновлення інформації про статистику, з подальшою можливістю перегляду її у вигляді кругових діаграм і гістограми індивідуально для кожного закладу харчування базуючись на отриманих відповідях від користувачів.

Блок-схема алгоритму методу оцінювання якості обслуговування наведена на рисунку 2.11.

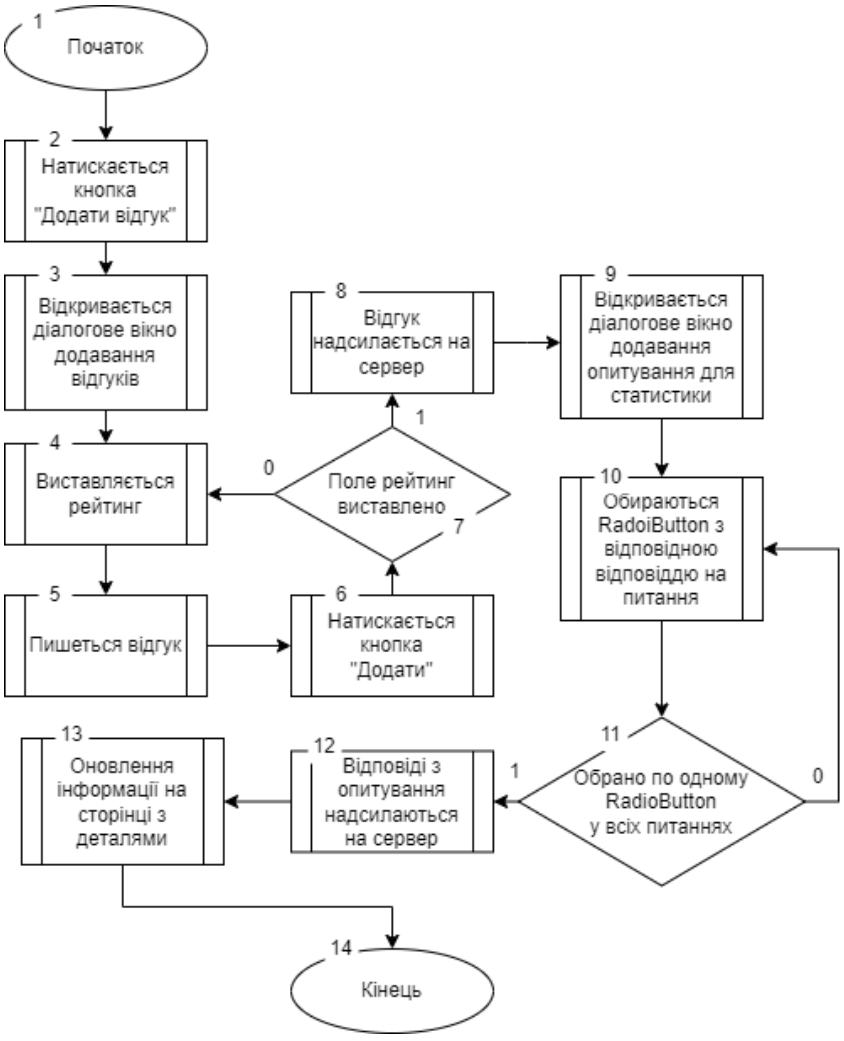



Рисунок 2.11 – Блок-схема алгоритму оцінювання якості обслуговування

Після того, як коментар буде додано і діалогове вікно, в якому додано коментар, буде закрито, відкриється наступне діалогове вікно для додавання екрану, на якому відображується детальна інформація про заклад, яка оновиться для того, щоб у списку з’явився коментар, який щойно був доданий користувачем.

### 2.5 Розробка методу бронювання місць у закладах харчування

Функція бронювання місць у закладах харчування є однією з найбільше затребуваних у додатках такого типу як «RestoBooking». Це дає можливість зекономити час користувача, не витрачаючи його на дзвінки для бронювання. Для цього необхідно обрати заклад, який сподобався, натиснути кнопку забронювати, заповнити дані про замовлення, а саме дату, час і кількість людей. Метод бронювання закладів харчування охоплює послідовність дій:

1. Користувач натискає кнопку «Забронювати» на екрані з деталями про заклад.
2. Обирається дата бронювання.
3. Обирається час від якого бронюється місце.
4. Обирається час до якого бронюється місце.
5. Обирається кількість людей на яку бронюється столик.
6. Натискається кнопка забронювати.
7. З трьох об’єктів типу Date, а саме: дата бронювання, час від якого бронюється і час до якого бронюється, створюється два об’єкти типу Date, а саме: дата бронювання з часом від якого бронюється і дата бронювання з кінцевим часом до якого бронюється.
8. Два об’єкти типу Date конвертуються в об’єкти типу Long, для подальшого зберігання в Cloud Firebase.
9. Бронювання відсилається на сервер

Блок-схема алгоритму бронювання закладів харчування зображена на рисунку 2.12.


Рисунок 2.12 – Блок-схема алгоритму бронювання закладів харчування

Після того, як місце буде заброньовано, діалогове вікно, в якому додавався коментар, буде закрито, і коли користувач перейде на екран «Мої бронювання», там буде відображено всі його актуальні бронювання у закладах і відповідно до цього списку додається заклад, який був щойно заброньований.

## 2.6 Розробка моделі Android-системи для бронювання і моніторингу якості обслуговування в закладах харчування

Існування зв'язків всередині системи та відносин між її елементами є однією з ключових характеристик будь-якої системи. Графічне подання системи здійснюється за допомогою уніфікованої мови моделювання (UML).

UML (Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації, документування і верифікації програмних систем. Це не мова програмування, а набір правил і стандартів для створення діаграм, які описують структуру і поведінку системи [25].

UML використовується в різних галузях, включаючи розробку програмного забезпечення, бізнес-аналітику, системне проектування та архітектуру. UML використовується для моделювання різних аспектів програмних систем, включаючи структуру, поведінку, взаємодію і функціональність. Це допомагає розробникам краще зрозуміти систему, передбачити можливі проблеми і уникнути помилок. Також, UML використовується для моделювання бізнес-процесів, вимог і систем. Це допомагає бізнес-аналітикам краще зрозуміти потреби бізнесу і розробити ефективні рішення. Ще, уніфікована мова моделювання використовується для моделювання складних систем, таких як мережі, системи управління та системи реального часу. Це допомагає системним інженерам краще зрозуміти систему і розробити ефективні архітектури.

Основні типи діаграм UML:

- діаграми структури, які описують статичну структуру системи і до яких відносяться діаграми класів, діаграми компонентів, діаграми розгортання та діаграми пакетів;

- діаграми поведінки, які описують поведінку системи і до яких відносяться діаграми станів, діаграми діяльності, діаграми послідовностей та діаграми кооперації;
- діаграми реалізації описують фізичну реалізацію системи, до них відносяться діаграми реалізації класів, діаграми компонентів, діаграми розгортання та діаграми автоматичних машин.

UML є потужним інструментом, який може допомогти розробникам, бізнес-аналітикам і системним інженерам краще зрозуміти і розробити складні системи.

Для візуального представлення взаємодії між акторами (користувачами) та варіантами використання використовуються діаграми прецедентів.

Діаграма прецедентів — Use-case diagram. Діаграма прецедентів — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання. Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Діаграми прецедентів відображають елементи моделі варіантів використання [26].

Діаграма прецедентів використовує 2 основних елементи:

1. Actor (учасник) — множина логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Учасником може бути людина, роль людини в системі чи інша система, підсистема або клас, які представляють щось поза сутністю.
2. Use case (прецедент) — опис окремого аспекту поведінки системи з точки зору користувача. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

На рисунку 2.13 показано модель Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування, для якої було розроблено систему компонентного аналізу з використанням діаграм прецедентів.

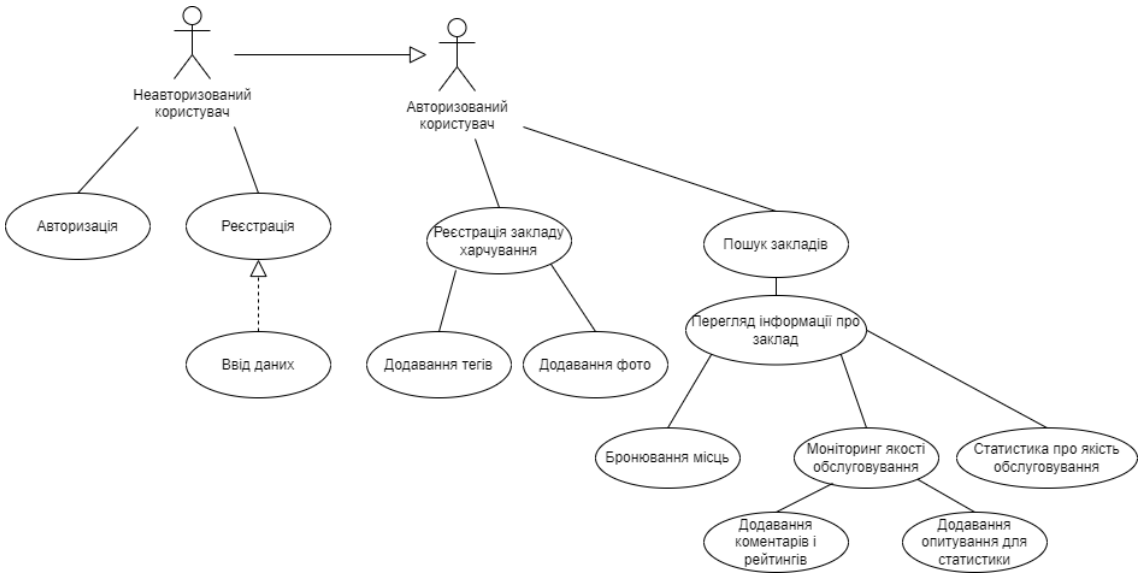



Рисунок 2.13 – Модель Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування

Перелік варіантів використання та їх короткий опис наведено в таблиці 2.1.

Таблиця 2.1 – Реєстр варіантів використання

| Актор | Прецедент | Опис |
|---|---|---|
| Неавторизований користувач | Авторизація | Введення логіну та паролю для здійснення входу в систему. |
| | Реєстрація | Введення та збереження особистих даних нового користувача для подальшої можливості авторизації. |
| Авторизований користувач | Реєстрація закладу харчування | Реєстрація нового закладу харчування у системі. |
| | Додавання тегів закладу | Додавання тегів закладу під час його реєстрації. |
| | Пошук закладу | Можливість знайти заклад відповідно до своїх потреб та бажань. |

Продовження табл. 2.1

| Актор | Прецедент | Опис |
|---|---|---|
| Авторизований користувач | Бронювання місць | Можливість забронювати місце у закладі. |
| | Додавання відгуків про заклад | Можливість додати відгук про заклад та оцінити якість обслуговування в ньому. |

Головний функціонал Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування полягає в спрощенні процесу бронювання місць, що дозволить економію часу користувачів на пошук закладу і на процес бронювання.

### 2.7 Висновки

У другому розділі було обґрунтовано вибір архітектурного паттерну MVVM, вибір Jetpack Compose для реалізації графічного інтерфейсу, а також розроблено методи, моделі й алгоритми роботи системи, зокрема, для реалізації базового функціоналу системи: додавання нового закладу харчування в систему, пошуку закладів харчування відповідно до обраного міста і обраних тегів, додавання відгуків про якість обслуговування і формування рейтингу закладів харчування та бронювання місць в обраних закладах.

# 3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ANDROID-СИСТЕМИ ДЛЯ БРОНЮВАННЯ МІСЦЬ ТА МОНІТОРИНГУ ЯКОСТІ ОБСЛУГОВУВАННЯ В ЗАКЛАДАХ ХАРЧУВАННЯ

## 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного продукту

Сьогодні технології розробки програмного забезпечення розвиваються з блискавичною швидкістю. Тому не дивно, що кількість мов програмування зростає теж високим темпом.

Вибір мови програмування для розробки програмного забезпечення є дуже важливим питанням, яке постає перед кожним початком розробки програми. При розробці мобільних додатків, призначених для запуску на смартфонах під управлінням ОС Android, пріоритетними є мови з відповідними функціями, такими як графічні інтерфейси, здатність обробляти дії користувача та можливість легко і швидко маніпулювати файлами. Було розглянуто найпоширеніші мови програмування, що використовуються при розробці мобільних Android-додатків, а саме Java та Kotlin.

Java – це широко використовувана об’єктно-орієнтована мова програмування та програмна платформа, яка працює на мільярдах пристроїв, включаючи ноутбуки, мобільні пристрої, ігрові консолі, медичні пристрої та багато інших. Правила і синтаксис Java засновані на мовах C і C++.

Однією з основних переваг розробки програмного забезпечення за допомогою Java є його переносимість. Після того, як написати код для програми на Java на ноутбуці, його дуже легко перемістити на мобільний пристрій. Коли в 1991 році Джеймсом Гослінгом із Sun Microsystems (пізніше придбаної Oracle) цю мову винайшов, головною метою була можливість «написати один раз і запустити будь-де».

Також важливо розуміти, що Java значно відрізняється від JavaScript. Javascript не потрібно компілювати, тоді як код Java потребує компілювання.

Крім того, Javascript працює лише у веб-браузерах, тоді як Java можна

запускати будь-де.

Нові та вдосконалені інструменти розробки програмного забезпечення надходять на ринок із надзвичайною швидкістю, витісняючи існуючі продукти, які раніше вважалися незамінними. У світлі цього постійного обороту довговічність Java вражає. Більш ніж через два десятиліття після свого створення Java все ще залишається найпопулярнішою мовою для розробки програмного забезпечення — розробники продовжують обирати її замість таких мов, як Python, Ruby, PHP, Swift, C++ та інших. Як результат, Java залишається важливою вимогою для конкуренції на ринку праці.

Щоб створити програму за допомогою Java, потрібно завантажити Java Development Kit (JDK), який доступний для Windows, macOS і Linux. Написану програму на мові програмування Java, потім компілятор перетворює програму на байт-код Java — набір інструкцій для віртуальної машини Java (JVM), яка є частиною середовища виконання Java (JRE). Байт-код Java працює без змін у будь-якій системі, яка підтримує JVM, що дозволяє виконувати ваш код Java будь-де.

Програмна платформа Java складається з JVM, Java API і повного середовища розробки. JVM аналізує та запускає (інтерпретує) байт-код Java. Java API складається з великого набору бібліотек, включаючи базові об’єкти, функції мережі та безпеки; Генерація розширюваної мови розмітки (XML); і веб-сервіси. Разом мова Java і програмна платформа Java створюють потужну, перевірену технологію для розробки корпоративного програмного забезпечення.

Kotlin — це мова програмування, яка була випущена в 2011 році компанією JetBrains, яка продає інтегровані середовища розробки (IDE) для мов програмування. Відтоді вона стала улюбленою мовою розробників і замінила Java у багатьох програмних проектах.

Створення Kotlin виникло після того, як провідний розробник Дмитро Джеметров шукав функції, які він не міг знайти в Java. Scala, інша мова, яка працює на віртуальній машині Java (JVM), була близькою до того, що він хотів, але її компіляція зайняла надто багато часу.

Джемеров хотів мову, яка мала б усі функції більш сучасних мов програмування, працювала б на JVM і компілювалася так само швидко, як Java. Тож він створив власну мову Kotlin [27].

Kotlin був розроблений як заміна Java в операційній системі Android. Через вісім років після випуску, у 2019 році, Google нарешті погодився з Джемеровим і більшістю розробників Android і оголосив, що Kotlin є кращою мовою для розробки додатків Android [28].

Kotlin розроблений для роботи на віртуальній машині Java і може працювати поруч із Java. Незважаючи на те, що Kotlin спочатку був розроблений спеціально для Android, він швидко поширився серед спільноти Java завдяки своїм функціям і з тих пір використовується для багатьох типів програм.

Основні переваги Kotlin наступні:

− Kotlin є лаконічним, заощаджуючи час, який розробник витрачає на написання шаблонного коду на Java. файл Java у файл Kotlin за допомогою лише сценарію;

− Kotlin не має накладних витрат на виконання, іноді додавання функцій до мови означає, що вона має більше накладних витрат, що знижує її продуктивність, але у Kotlin не так;

− Kotlin має велику спільноту, якщо коли-небудь виникнуть проблеми при написані коду, то можна легко знайти інших розробників, які допоможуть вам на форумах кодування та в соціальних мережах;

− Здійснювати асинхронні виклики мережі та бази даних у Java незграбно та болісно, але Kotlin має Coroutines, які роблять асинхронне програмування простим і ефективним;

− null у Java може призвести до збою програми, якщо до цього не підготуватися, але у Kotlin можна додати простий оператор до змінних, які можуть бути нульовими, щоб запобігти цим збоям.

Наявність мобільного зв’язку є обов’язковою умовою для більшості підприємств, оскільки зараз більшість людей мають доступ до Інтернету через мобільні телефони. На Android припадає понад 70% частки ринку мобільних

телефонів, тож навіть якби Kotlin був лише для розробки Android, розробники Kotlin мали б великий попит.

Результати порівняння розглянутих мов програмування за обраними критеріями наведені у таблиці 3.1.

Таблиця 3.1 – Порівняльні характеристики мобільних додатків

| Мова програмування / Критерій | Java | Kotlin |
|---|---|---|
| Extensions | - | + |
| Можливість розробки для ОС Android | + | + |
| Об’єктно-орієнтованість | + | + |
| Null safeety | - | + |
| Підсумковий результат | 2 | 4 |

Таким чином, можна зробити висновок, що Kotlin є однією з кращих мов програмування серед порівнюваних мов і відповідає всім вимогам для розробки Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування «RestoBooking».

### 3.2 Розробка програмного модуля реєстрації користувача

Перш ніж перейти до розробки модулів програми, необхідно розробити модуль реєстрації користувача. Реєстрація у мобільний додаток «RestoBooking» відбувається через електронну пошту, яку при авторизації користувач має вказати. Також користувач повинен ввести своє ім’я та прізвище, яке буде відображатись на екрані персональної інформації, а також, коли користувач буде додавати відгуки.

Також користувачеві потрібно ввести пароль. Для того, щоб впевнитися, що користувач вводить не абиякий пароль, а такий, який він зможе ввести знову при авторизації, тобто не забуде його одразу після введення, було додано ще одне поле, у якому користувач повинен ввести пароль повторно. Також доступна функція зробити пароль видимим. Метод, який добавляє нового користувача в систему, зображено на рисунку 3.1.

```
override suspend fun firebaseSignUpNewUser(user: CreateUser): Result<Unit> =
    try {
        val result = auth.createUserWithEmailAndPassword(user.email, user.password).await()
        val profileUpdates = UserProfileChangeRequest.Builder()
            .setDisplayName(user.nameSurname)
            .build()
        result.user?.updateProfile(profileUpdates)?.await()
        Result.success(Unit)
    } catch (e: Exception) {
        Result.failure(e)
    }
```

Рисунок 3.1 – Реєстрація нового користувача

Першим кроком проводиться реєстрація нового користувача. Після того, як цей запит пройде успішно, для цього користувача встановлюється ім’я та прізвище, яке було введено користувачем.

Метод помічено модифікатором suspend. Модифікатор suspend у Kotlin використовується для позначення функцій, які можуть бути призупинені і відновлені в будь-який момент. Це дозволяє виконувати такі завдання, як робота з зовнішніми ресурсами, такими як мережа або база даних, або виконання довгих обчислень.

Функції, позначені модифікатором suspend, можуть викликатися тільки з інших функцій, позначених тим же модифікатором, або з корутини. Це пов'язано з тим, що виконання функції suspend може бути призупинено в будь-який момент, і компілятору потрібно знати, як відновити виконання функції, яка її викликала.

У цьому випадку робиться два запити: перший на додавання нового

користувача і другий на оновлення його імені та прізвища. Опрацювання цих запитів займе деякий час. Саме тому використовуються suspend модифікатор і корутини, які дають можливість легко і зручно керувати асинхронними процесами.

### 3.3 Розробка програмного модуля реєстрації нового закладу харчування

Розробка модулю реєстрації нового закладу в системі є одним з найважливіших модулів у системі. Вся реєстрація нового закладу розподілена на чотири кроки, щоб користувачеві було простіше та зручніше відслідковувати великий об’єм інформації. Після кожного заповнення певного екрану, інформація зберігається у ViewState того view, у якого заповнювались поля.

У Jetpack Compose ViewState – це об'єкт, який представляє стан UI. Він містить всі дані, необхідні для відображення UI, а також інформацію про те, як відображати ці дані.

ViewState можна використовувати для зберігання будь-яких даних, які необхідні для відображення UI. Наприклад, він може зберігати список елементів, значення текстового поля або стан кнопки.

ViewState також може містити інформацію про те, як відображати ці дані. Наприклад, він може містити інформацію про стиль, розмір або розташування елементів UI.

ViewState зазвичай використовується в поєднанні з ViewModel. ViewModel - це клас, який відповідає за зберігання та обробку даних для UI. ViewModel може оновлювати ViewState, щоб відобразити зміни даних.

ViewState робить розробку UI більш ефективною та гнучкою. Він допомагає відокремити код UI від коду даних. Це також допомагає зробити UI більш реактивним, оскільки ViewState може оновлюватися динамічно.

Далі після того, як всі поля були заповені, а також було додано хоча б одне фото, викликається метод registerFoodEstablishment, який на вхід приймає foodEstablishment який йому необхідно додати в систему.

На рисунку 3.2 зображено блок-схему функції registerFoodEstablishment, яка додає нові заклади харчування в систему.

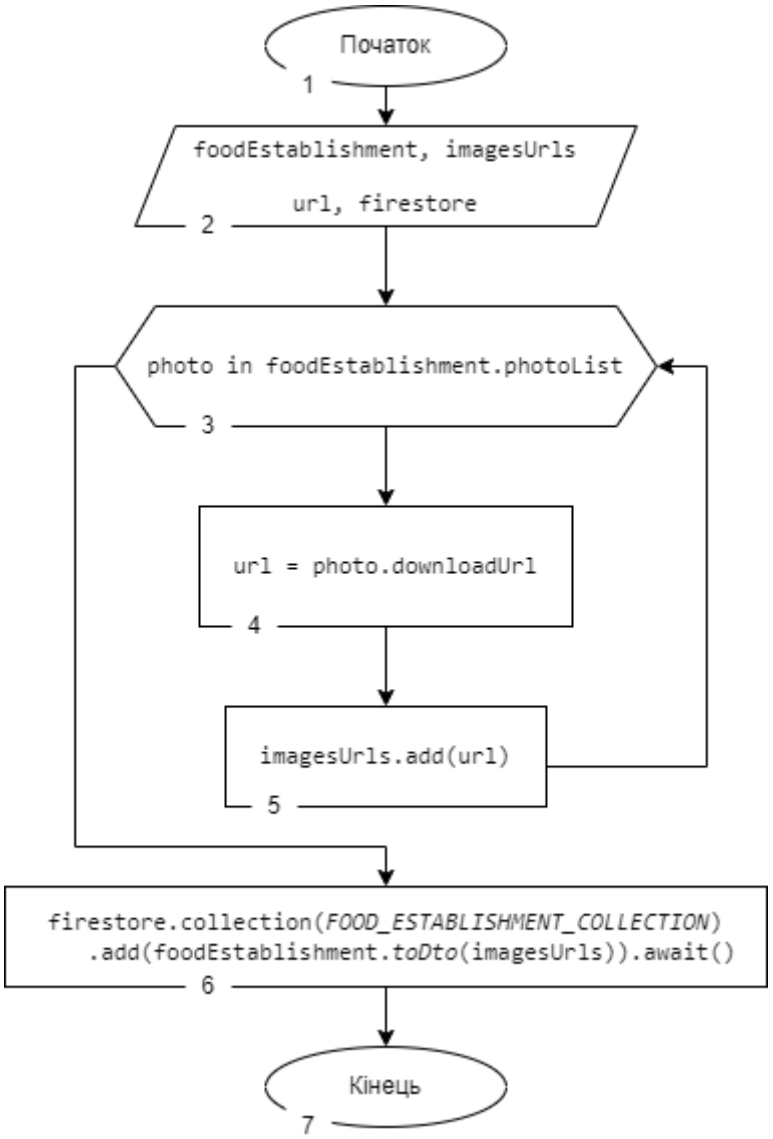



Рисунок 3.2 – Блок-схема функції registerFoodEstablishment

Фотографії зберігаються у вигляді списку з посиланнями downloadUrl, які ведуть до Firebase Storage. Саме за цими посиланнями завантажуються усі фотографії.

Firebase Storage – це сервіс зберігання об'єктів, який надає розробникам мобільних і веб-додатків можливість легко і надійно зберігати і обробляти зображення, відео, документи та інші файли.

Firebase Storage використовує інфраструктуру Google Cloud Storage, яка

забезпечує високу доступність, масштабованість і безпеку. Сервіс підтримує різні типи файлів, включаючи зображення, відео, документи, архіви та інші.

Firebase Storage можна використовувати для наступних цілей:

1. Зберігання і обробка контенту, створеного користувачами. Наприклад, Firebase Storage можна використовувати для зберігання фотографій, відео та інших файлів, які користувачі завантажили в ваше додаток.
2. Розміщення статичного вмісту. Наприклад, Firebase Storage можна використовувати для розміщення зображень, CSS-файлів та інших статичних ресурсів вашого додатку.

Переваги використання Firebase Storage:

1. Легкість використання. Firebase Storage має простий і інтуїтивний API, який дозволяє швидко і легко інтегрувати сервіс у додаток.
2. Надійність. Firebase Storage використовує інфраструктуру Google Cloud Storage, яка забезпечує високу доступність, масштабованість і безпеку.
3. Безпека. Firebase Storage підтримує різні рівні безпеки, включаючи авторизацію на основі ролей і шифрування.

### 3.4 Розробка програмного модуля додавання коментарів

Модуль додавання коментарів є важливою складовою додатку «RestoBooking». Його функціонал виконує метод addComment, який у якості аргументів отримує foodEstablishmentId у вигляді String, це ID закладу харчування по якому буде вестися пошук якому саме закладу добавити коментар. Також у аргументах цієї функціє є commentText тип даних якого String, йе власне і є сам текст коментаря, який може бути порожнім, і в такому випадку, з допомогою конструкції commentText.ifEmpty { null } передається null, якщо користувач не заповнив поле коментаря. І останнє поле, яке є у аргументах даного методу це rating типу int, який представляє собою рейтинг, який обрав користувач на діалоговому вікні. На рисунку 3.4 зображено фрагмент коду метода, який шукає заклад, якому повинен додатися написаний користувачем коментар.

```
val currentUser = userRepository.currentUser?.displayName
try {
    val collection = firestore.collection(FOOD_ESTABLISHMENT_COLLECTION)
    val task =
        collection.whereEqualTo(FoodEstablishment::id.name, foodEstablishmentId).get()
            .await()
    val idInDatabase = task.documents[0].id
    val foodEstablishmentRef = collection.document(idInDatabase)

    val document = foodEstablishmentRef.get().await()
    val foodEstablishment = document.toObject(FoodEstablishmentDto::class.java)?.toDomain()
    val currentComments = foodEstablishment?.comments?.toMutableList()
```

Рисунок 3.3 – Фрагмент коду, який шукає заклад, якому повинен додатися написаний користувачем коментар

Цей код шукає заклад в базі даних за унікальним id, який генерується для закладу при його реєстрації. Потім дістається екземпляр обраного закладу харчування і звідти береться поточний список коментарів закладу.

Наступим кроком формується об’єкт нового коментаря і додається до масиву поточних коментарів. Фрагмент цього коду зображено на рисунку 3.4.

```
val currentDate = Calendar.getInstance().timeInMillis
val newComment = Comment(
    author = currentUser ?: "",
    commentText = commentText.ifEmpty { null },
    rating = rating,
    dateAdded = currentDate
)
currentComments?.add(newComment)
```

Рисунок 3.4 – Фрагмент коду, який створює об’єкт коментаря і додає його до поточного масиву коментарів

Фінальним кроком є оновлення масиву коментарів, який присвоюється об’єкту заклада харчування, котрий було раніше знайдено за ID. У випадку, якщо коментар було додано успішно, повертається об’єкт типу Result.success(Unit),

але якщо виникне якась помилка, то повернеться Result.failure(Exception("Comment was not added successfully")). Фрагмент коду, який відправляє коментар на сервер, зображено на рисунку 3.5.

```
val images: List<String> =
    foodEstablishment?.photoList?.map { it.uri.toString() } ?: emptyList()
val updatedFoodEstablishment: FoodEstablishmentDto? =
    foodEstablishment?.copy(comments = currentComments ?: emptyList())?.toDto(images)

if (updatedFoodEstablishment != null) {
    foodEstablishmentRef.set(updatedFoodEstablishment).await()
    return@withContext Result.success(Unit)
}
Result.failure(Exception("Comment was not added successfully")) ^withContext
```

Рисунок 3.5 – Фрагмент коду, який відправляє коментар на сервер

Для того, щоб дізнатись ім’я авторизованого на даний момент користувача, який додає даний відгук, з userRepository завантажується displayName поточного авторизованого користувача, який зберігається у даному репозиторії. Для того, щоб збільшити час додавання коментаря, робиться запит поточного часу з допомогою Calendar.getInstance(), і звідси береться змінна timeInMillis, яка повертає значення типу Long.

## 3.5 Розробка програмного модуля завантаження усіх закладів харчування

Для завантаження усіх закладів харчування, які відповідають заданим користувачем параметрам, використовується метод fetchFoodEstablishments з репозиторія FoodEstablishmentRepository. У якості аргументів метод приймає два значення. Перше з них є city типу даних String, і воно може бути порожнім, а друге значення – це масив типу String, який не може бути порожнім і має містити хоча б один елемент.

На рисунку 3.6 зображено блок-схему алгоритму роботи функції fetchFoodEstablishments, яка завантажує заклади харчування.

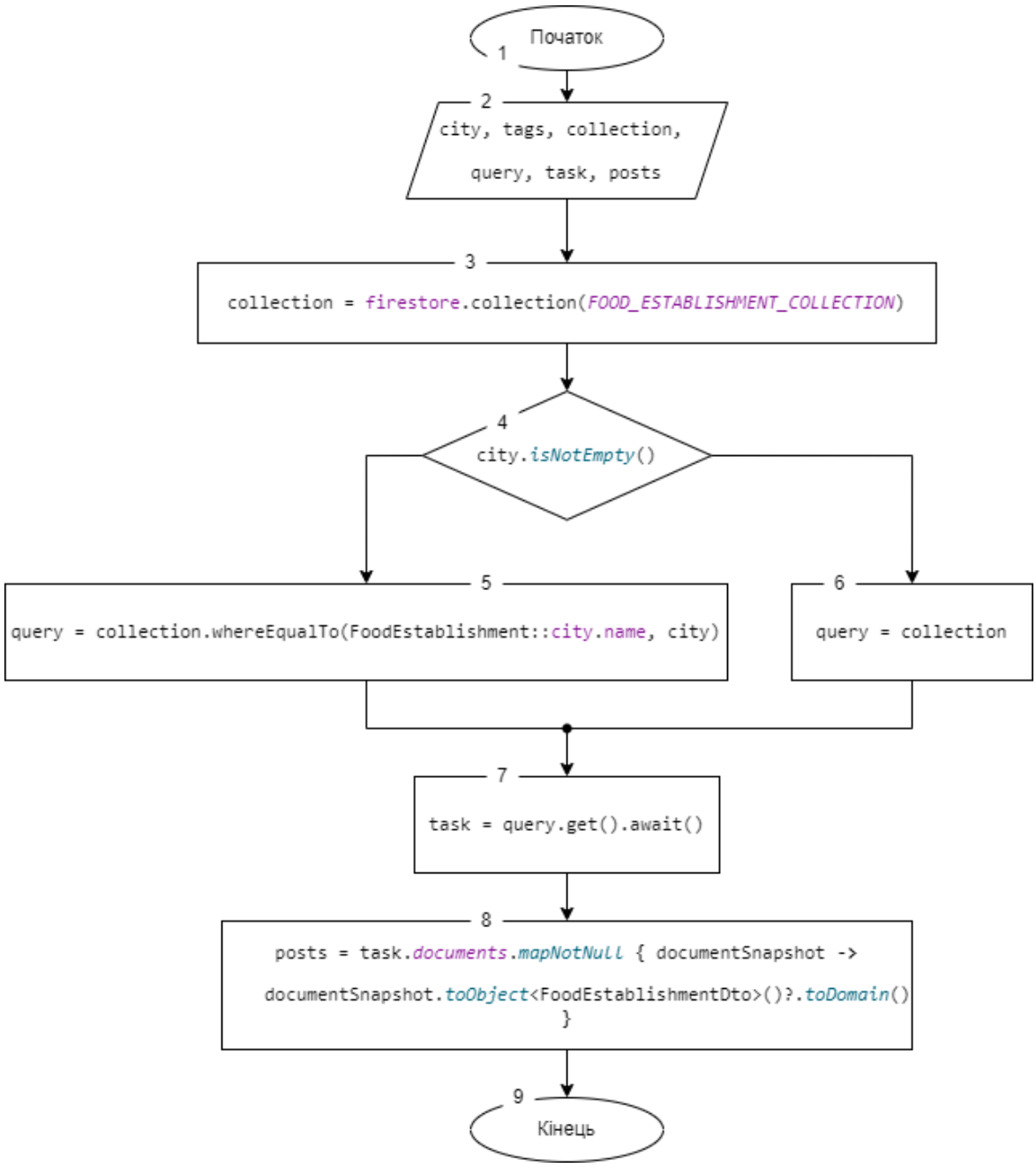



Рисунок 3.6 – Блок-схема алгоритму роботи функції fetchFoodEstablishments

Також у функції використовується extension для перетворення моделі з моделі data рівня у модель domain рівня FoodEstablishmentDto.toDomain(). Приклад коду описаного методу наведено на рисунку 3.7.

```
fun FoodEstablishmentDto.toDomain(): FoodEstablishment {
    var totalRatingCount = 0.0f
    comments.forEach { it: Comment
        totalRatingCount += it.rating
    }
    val rating: Float = totalRatingCount / comments.size
    return FoodEstablishment(
        id = this.id,
        name = this.name,
        foodEstablishmentType = this.foodEstablishmentType,
        address = this.address,
        description = this.description,
        photoList = this.photoUrlList.mapIndexedNotNull { index, uri ->
            Photo(
                index,
                Uri.parse(uri)
            )
        },
        ownerName = this.ownerName,
        city = this.city,
        selectedTimeFrom = this.selectedTimeFrom,
        selectedTimeTo = this.selectedTimeTo,
        phoneForBooking = this.phoneForBooking,
        tags = this.tags,
        comments = this.comments,
        rating = rating
    )
}
```

Рисунок 3.7 – Метод перетворення моделі FoodEstablishmentDto з моделі data рівня у модель domain рівня FoodEstablishment

Саме тут формується рейтинг для закладу харчування за методом знаходження середнього арифметичного серед усіх доданих відгуків, а також формуються об’єкти Photo, оскільки з серверу приходять посилання на фотографії на сервері.

### 3.6 Розробка програмного модуля для завантаження закладу харчування за ID

Після того, як користувач введе дані для пошуку закладів харчування, знайдені заклади завантажаться та покажуться у вигляді списку. Але на екрані користувачеві доступними є лише основні дані про заклад, без можливості зробити бронювання, додати відгук чи переглянути додаткову інформацію про

заклад. Для цього необхібно завантажити деталі закладу за ID, який передається на PdpScreen. Частина NavHostBuilder, яка здійснює навігацію на PdpScreen та передає ID обраного закладу харчування у аргументах, наведена на рисунку 3.8.

```
composable(
    route = "${AppDestinations.Pdp.route}/{$ID_ARG}",
    arguments = listOf(
        navArgument(ID_ARG) { type = NavType.StringType }),
    enterTransition = { this: AnimatedContentTransitionScope<NavBackStackEntry>
        fadeIn(
            animationSpec = tween( durationMillis: 700)
        )
    },
    exitTransition = { this: AnimatedContentTransitionScope<NavBackStackEntry>
        fadeOut(
            animationSpec = tween( durationMillis: 700)
        )
    }) { this: AnimatedContentScope  it: NavBackStackEntry
    PdpScreen(
        navigateBack = {
            navController.popBackStack()
        }
    )
}
```

Рисунок 3.8 – Навігація на PdpScreen та передача ID обраного закладу харчування у аргументах

Наступним кроком у PdpScreenViewModel з SavedStateHandle дістається Id закладу, яке раніше передавалося в аргументи з допомогою методу checkNotNull(savedStateHandle[ID_ARG]). Усі запити на сервер відбуваються на CoroutineContext Dispatcher IO. Dispatcher IO — це CoroutineDispatcher, призначений для обробки системних операцій введення/виведення (Input/Output) у коді.

Цей диспетчер використовує пул потоків, окремий від пулу потоків, який використовує диспетчер за замовчуванням. Це означає, що співпрограми, запущені на диспетчері введення-виведення, не блокуватимуть головний потік

або інші співпрограми, запущені на диспетчері за замовчуванням.

За замовчуванням кількість потоків у пулі потоків диспетчера вводу-виводу встановлюється або на 64, або на кількість ядер ЦП, доступних системі, залежно від того, що більше. Однак можна змінити кількість потоків у пулі, змінивши значення системної властивості kotlinx.coroutines.io.parallelism.

Метод, який завантажує заклад харчування за його ID, зображено на рисунку 3.9.

```
override suspend fun getFoodEstablishmentById(id: String): Result<FoodEstablishment> =
    withContext(Dispatchers.IO) { this: CoroutineScope
        try {
            val collection = firestore.collection(FOOD_ESTABLISHMENT_COLLECTION)
            val task = collection.whereEqualTo(FoodEstablishment::id.name, id).get().await()
            val posts = task.documents.mapNotNull { documentSnapshot ->
                documentSnapshot.toObject<FoodEstablishmentDto>()?.toDomain()
            }
            Result.success(posts[0]) ^withContext
        } catch (e: Exception) {
            Result.failure(e) ^withContext
        }
    }
```

Рисунок 3.9 – Завантаження закладу харчування  по його ID

На початку вказується колекція елементів, до якої буде здійснено запит. У цьому випадку назва колекції FOOD_ESTABLISHMENT_COLLECTION, яка передається у firestore.collection(). Далі вказується що пошук буде проводитись саме за полем ID і за самим значенням ID. Після того, як всі елементи отримуються, вони перетворюються з моделі FoodEstablishmentDto, яка зберігається на рівні data, у модель FoodEstablishment рівня domain. І останнім кроком повертається знайдений елемент закладу харчування, який у подальшому буде продемонстровано користувачеві.

За відображення деталей закладу харчування відповідає Composable метод FoodEstablishmentDetailsView (див. рисунок 3.10), який в аргументи приймає всі необхідні значення, які йому потрібно відобразити.

```
FoodEstablishmentDetailsView(
    name = uiState.foodEstablishment?.name ?: "",
    rating = uiState.foodEstablishment?.rating ?: 0.0f,
    tags = uiState.foodEstablishment?.tags ?: emptyList(),
    city = uiState.foodEstablishment?.city ?: "",
    address = uiState.foodEstablishment?.address ?: "",
    phoneForBooking = uiState.foodEstablishment?.phoneForBooking ?: "",
    workingTime = viewModel.getWorkingHours(),
    description = uiState.foodEstablishment?.description ?: "",
    comments = uiState.foodEstablishment?.comments?: emptyList()
) {
    viewModel.showAddCommentDialog( value: true)
}
```

Рисунок 3.10 – Завантаження закладу харчування  за його ID

Функція також обробляє кліки на кнопку про додавання коментаря, яка викликає метод viewModel.showAddCommentDialog(true), що змінює State showAddCommentDialog на true, тим самим запускаючи діалогове вікно додавання коментарів.

### 3.7 Розробка програмного модуля для бронювання місць у закладах харчування

Коли користувач знаходить заклад, який він хотів би забронювати, у нього є можливість здійснити бронювання. Після того, як користувач натисне кнопку «Забронювати», йому відобразиться діалогове вікно, у якому необхідно обрати кількість осіб, для яких здійснюється бронювання, дату, на яку бронюється місце, а також часові межі використання забронюваного місця. Крім того, є можливість залишити додатковий коментар до формування замовлення, для попередньої і швидкої комунікації із закладом з приводу побажань щодо замовлення.

На рисунку 3.11 зображено блок-схему функції addReservation, яка додає бронювання місця у закладі харчування.

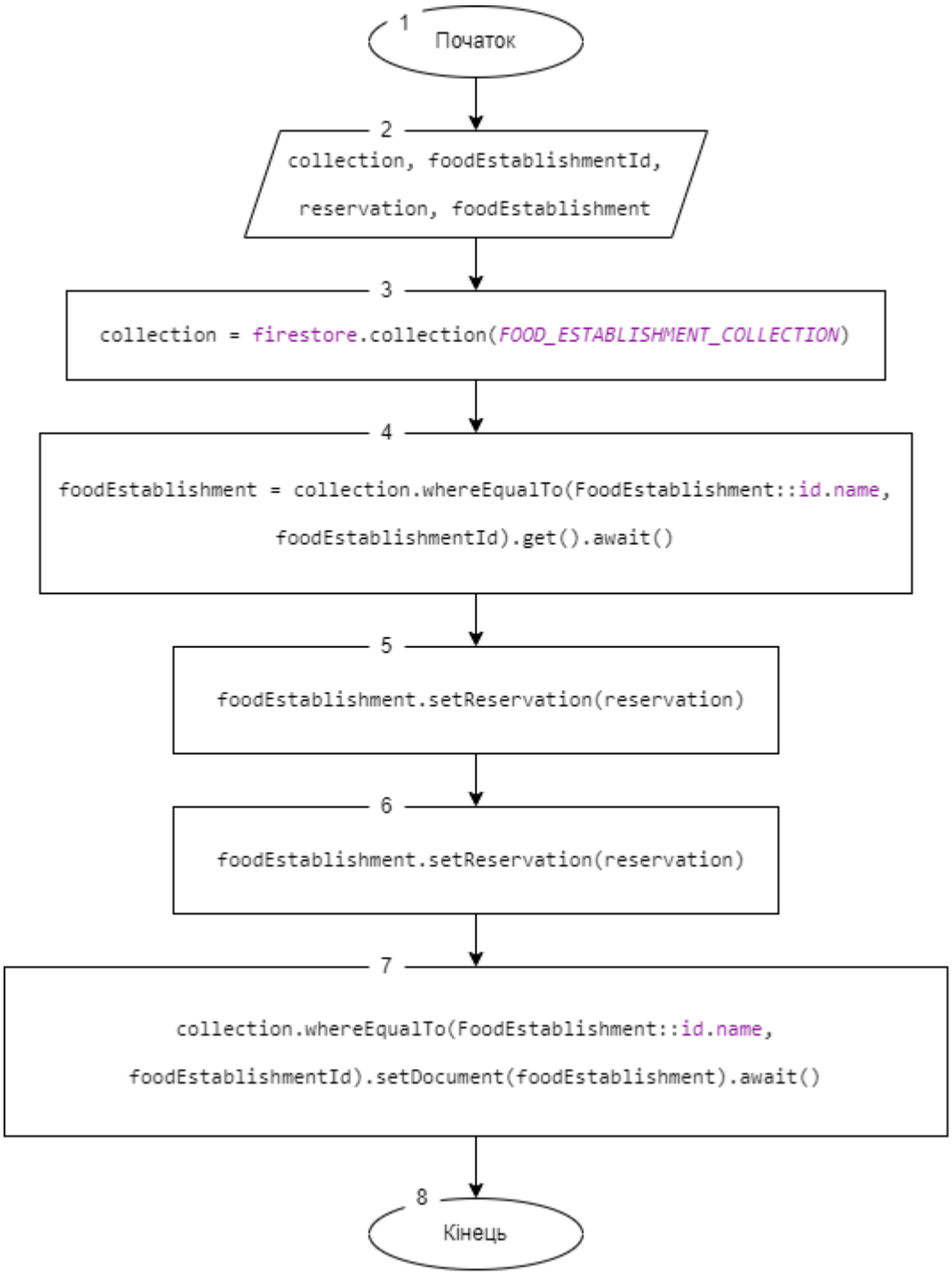



Рисунок 3.11 – Завантаження закладу харчування  за його ID

Заклад харчування, у якому має відбутися бронювання, шукається за ID, щоб його було легко оновити на сервері. Наступним кроком локально знайденому об’єкту встановлюється бронювання, а саме встановлюється дата бронювання та часові межі експлуатації, а також вказується кількість людей, для яких бронюють місця. Останній крок – це заміна старого об’єкта

FoodEstablishment на новий, який вже містить щойно додане бронювання.

### 3.8 Висновки

У третьому розділі було здійснено обґрунтований вибір засобів програмування та технологій, які використано при розробці програмного продукту, обрано середовище розробки, а також розроблено усі модулі програми.

Було обрано мову програмування Kotlin, інтегроване середовище розробки Android Studio і для написання графічного інтерфейсу користувача – Jetpack Compose. Вибрано систему для зберігання закладів харчування – Cloud Firestore, а для зберігання фото – Firebase Storage.

Розроблено Android-систему «RestoBooking» для бронювання місць і моніторингу якості обслуговування у закладах харчування, яка включає модуль реєстрації користувача; модуль реєстрації нового закладу харчування; модуль додавання коментарів; модуль завантаження усіх закладів харчування; модуль завантаження закладу харчування за ID; модуль бронювання місць у закладах харчування.

# 4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

## 4.1 Аналіз методів тестування

Тестування програмного забезпечення – це процес проведення експериментів для виявлення помилок і дефектів у програмі. Це допомагає переконатися, що програмне забезпечення працює правильно, відповідає вимогам та очікуванням користувачів, а також працює надійно та безпечно.

Перші програмні системи були розроблені в рамках програми для задоволення потреб науково-дослідних програм та Міністерства оборони. Тестування таких продуктів було суворо формалізоване, а всі процедури тестування, дані та результати фіксувалися. Тестування було розбите на окремі процеси, що починалися після завершення кодування, але зазвичай виконувалися одним і тим же персоналом [29].

Раніше, багато уваги приділялося "комплексному" тестуванню. Воно повинно проводитися з використанням усіх шляхів у коді або якомога більшої кількості вхідних даних. Було відзначено, що за таких умов неможливо повністю протестувати програмне забезпечення. Це пов'язано з тим, що, по-перше, кількість можливих вхідних даних занадто велика, по-друге, велика кількість шляхів, по-третє, важко знайти проблеми в архітектурі або специфікації. З цих причин "вичерпне" тестування було відкинуто і визнано теоретично неможливим.

Потім, тестування програмного забезпечення розглядалося як "процес, спрямований на демонстрацію правильності продукту" або "діяльність з перевірки правильності поведінки програмного забезпечення". У програмній інженерії, яка тоді ще перебувала в зародковому стані, верифікація програмного забезпечення визначалася як "доказ правильності". Хоча в теорії ця концепція була багатообіцяючою, на практиці вона забирала багато часу і не охоплювала всіх аспектів тестування. Доказ правильності вважався неефективним методом тестування програмного забезпечення. Однак у деяких випадках доказ правильності все ще використовується сьогодні, наприклад, при приймальному

тестуванні; наприкінці 1970-х років вважалося, що тестування – це запуск програми для пошуку помилок, а не для того, щоб довести, що вона працює. Успішне тестування - це тестування, яке виявляє раніше невідомі проблеми. Цей підхід є протилежним до попереднього. Ці два визначення представляють "парадокс тестування" і базуються на двох діаметрально протилежних твердженнях. З одного боку, тестування може підтвердити, що продукт працює добре; з іншого боку, тестування виявляє програмні помилки і показує, що продукт не працює.

Тестування, пов'язане з аналізом результатів розробки програмного забезпечення, називається статичним тестуванням. Воно включає перевірку програмного коду, моніторинг і верифікацію програми без її запуску на комп'ютері. Тестування, пов'язане з роботою програмного продукту, називається динамічним тестуванням. Динамічне і статичне тестування доповнюють одне одного.

На етапі статичного тестування перевіряються всі документи, що є результатом життєвого циклу програмного забезпечення. Сюди входять технічне завдання, специфікації та вихідний код програми на мовах програмування. Вся документація аналізується на відповідність стандартам програмування. За результатами статичних перевірок визначається, наскільки програма відповідає встановленим стандартам і вимогам замовника. Усунення неточностей і помилок у документації є гарантією високої якості вироблених програмних засобів.

Динамічні методи використовуються безпосередньо в процесі виконання програми. Коректність роботи програмного інструменту перевіряється за допомогою різних тестів і підготовлених наборів вхідних даних. Під час кожного запуску тесту збираються та аналізуються дані про помилки та збої в роботі програми.

Виділяють наступні типи тестування:

– метод білої скриньки;
– метод чорної скриньки;
– метод сірої скриньки.

Тестування "чорної скриньки" - це процес тестування системи або програмного забезпечення без попереднього знання того, як вона працює зсередини. Це включає в себе не тільки незнання самого вихідного коду, але й відсутність будь-якої проектної документації, пов'язаної з програмним забезпеченням. Тестувальник просто вводить дані і отримує результати так само, як і кінцевий користувач [30].

Мета тестування "чорної скриньки" - усунути упередження, що виникають через те, що користувач вже знає про програмне забезпечення, і дозволити йому взаємодіяти з програмним забезпеченням у більш природний спосіб, ніж зазвичай. У цій методології люди, відповідальні за проведення тестування, відрізняються від людей, які розробляють програмне забезпечення.

Тестування білого ящика оцінює внутрішню структуру програми та її код. Тестувальник повинен володіти достатніми знаннями структури коду, щоб переконатися, що всі операції виконуються правильно відповідно до специфікації. Тестування білого ящика допомагає виявити логічні, друкарські та дизайнерські помилки. За допомогою цього методу перевіряється, що всі цикли виконуються в межах своїх границь, що всі незалежні шляхи модулів використовуються хоча б один раз, і що жодне логічне рішення не є помилковим.

Метод сірого ящика - це спеціальний метод тестування, який поєднує в собі особливості тестування білого та чорного ящиків. Для виконання тесту не обов'язково знати всю структуру коду, достатньо мати знання про алгоритми поведінки та архітектуру програми. Іншими словами, тест відбувається на стороні користувача, але враховує відомі дані. Це зменшує ймовірність пропустити якийсь із тестових кейсів [31].

Отже, після аналізу кожного з методів тестування, визначення їхніх переваг і недоліків, було прийнято рішення використати метод «чорної скриньки». Використання саме цього методу, дасть можливість виявити помилки у функціонуванні мобільного додатку чи в роботі якогось з алгоритмів вже на ранній стадіїї.

## 4.2 Тестування Android-системи для бронювання місць і моніторингу якості обслуговування у закладах харчування

Коли користувач відкриває мобільний Android-додаток «RestoBooking», перший екран, який він бачить, – це традиційний для більшості мобільних додатків Splash screen (див. рисунок 4.1).


Рисунок 4.1 – Splash screen додатку «RestoBooking»

Якщо користувач не авторизувався раніше, він перейде на екран авторизації, де зможе ввести свою електронну пошту, а також пароль, який вказував при реєстрації, для подальшої роботи з додатком. Екран авторизації зображено на рисунку 4.2.

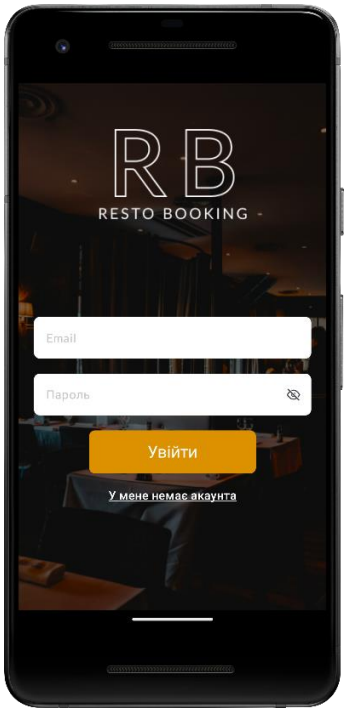



Рисунок 4.2 – Екран авторизації

Якщо користувач вперше відкриває додаток, у нього є можливість зареєструватися, натиснувши «У мене немає акаунта», після чого відкриється екран реєстрації (див. рисунок 4.3).

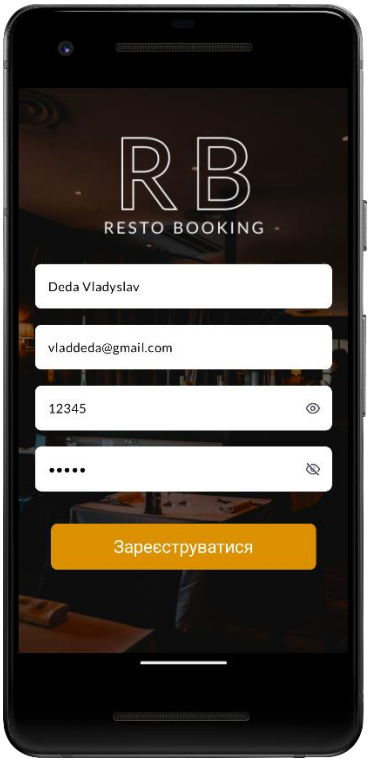



Рисунок 4.3 – Екран реєстрації

Для перевірки, чи успішно було зареєстровано нового користувача, можна увійти на консоль Firebase Auth і пересвідчитися, що у списку користувачів з'явився щойно доданий користувач (див. рисунок 4.4).

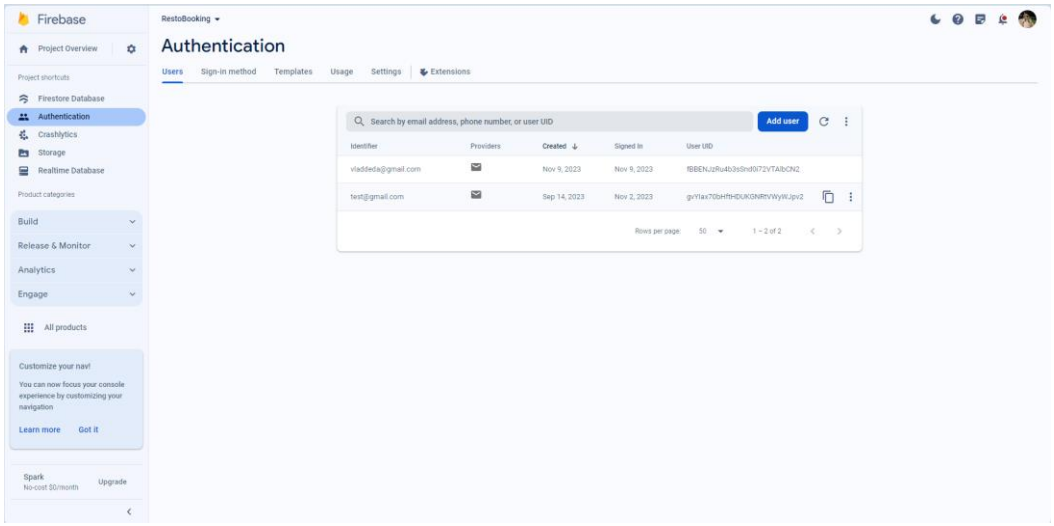



Рисунок 4.4 – Firebase Auth з новим користувачем у списку

Після того, як користувач авторизується або ж зареєструється успішно, він переходить на головний екран додатку, який зображено на рисунку 4.5. Тут є можливість знайти заклад харчування.

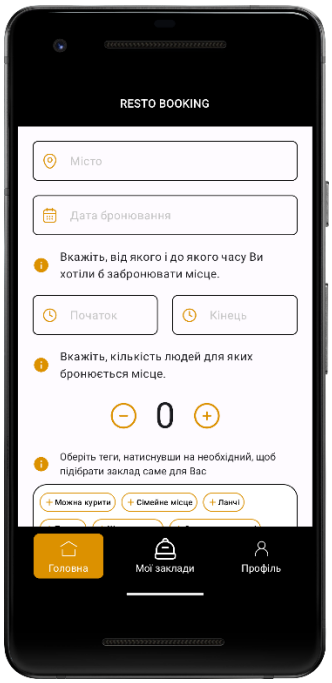



Рисунок 4.5 – Головний екран програми

Коли ж користувач вводить місто для пошуку, а також обирає теги, які мають відповідати закладу, який він шукає, кнопка пошук стане активною і користувач переходить на екран відображення результатів пошуку, який зображено на рисунку 4.6.

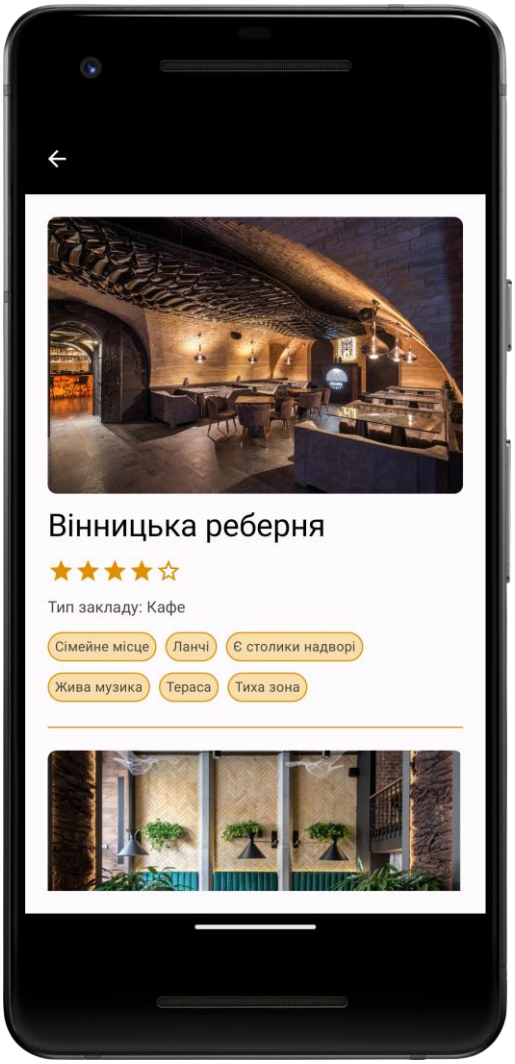



Рисунок 4.6 – Екран відображення результатів пошуку

На екрані відображення результатів пошуку, продемонстровано лише основну інформацію про заклад, а саме головне фото, назву, рейтинг закладу на основі відгуків, тип закладу, а також теги закладу. Для того, щоб переглянути детальну інформацію про заклад, необхідно натиснути на нього, після чого відкриється екран, на якому відображено всю основну інформацію про заклад. Екран з детальною інформацією зображено на рисунку 4.7.

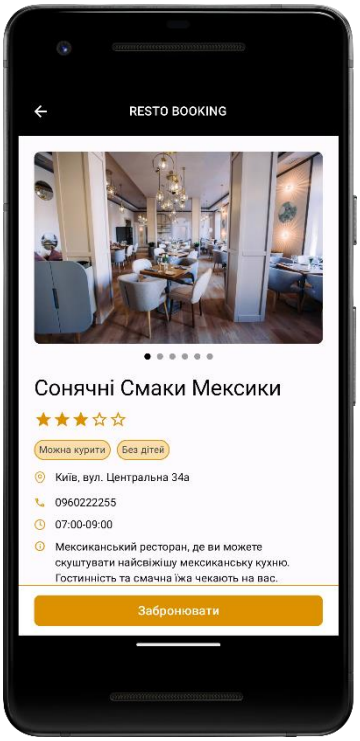



Рисунок 4.7 – Екран відображення детальної інформації про заклад

Для того, щоб переглянути коментарі про заклад, які залишили інші користувачі, необхідно прогорнути стрічку вниз, де буде відображено весь список коментарів про заклад (див. рисунок 4.8).

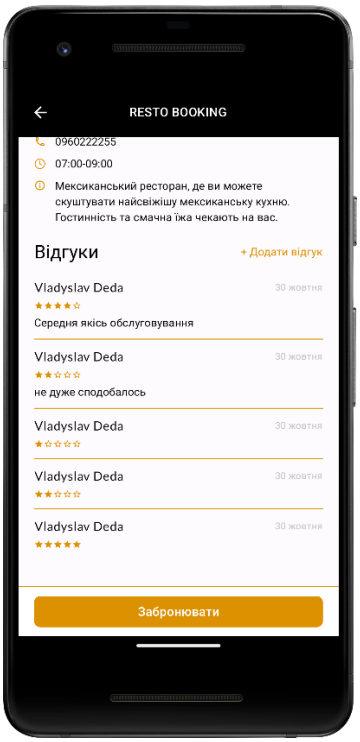



Рисунок 4.8– Список коментарів

Також на екрані деталей доступна кнопка «+ Додати коментар», натиснувши на яку, користувач має змогу відкрити діалогове вікно для додавання коментарів, як зображено на рисунку 4.9.

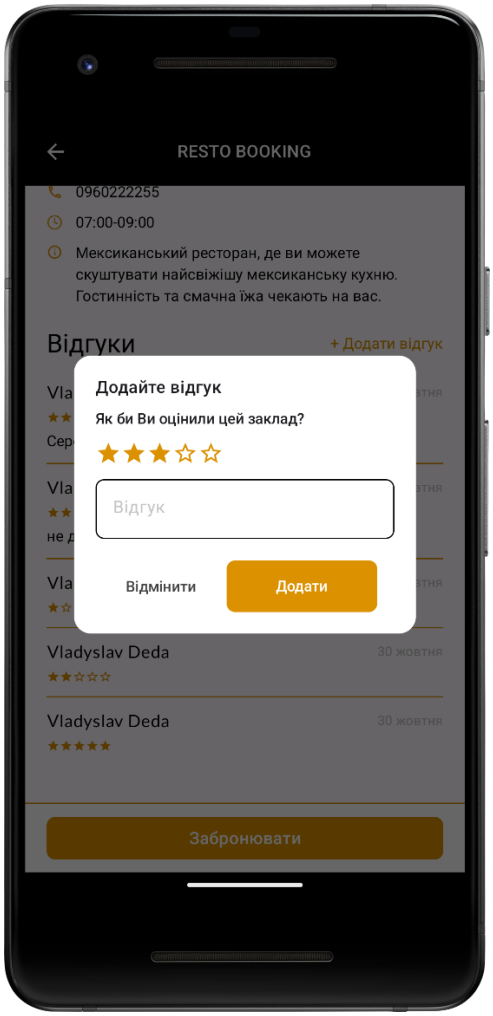



Рисунок 4.9– Список коментарів

Натиснувши кнопку «Забронювати», користувач має можливість забронювати місце у обраному закладі. Так як вся основна інформація про бронювання вводиться на головному екрані для зручного й швидкого пошуку закладів, на екрані бронювання, який зображено на рисунку 4.10, користувачеві надається можливість залишити коментар до бронювання з побажаннями. Коментар обов’язково побачить адміністратор закладу перед тим, як завершити бронювання. Відповідно, щоб здійснити бронювання, потрібно натиснути кнопку «Завершити бронювання».

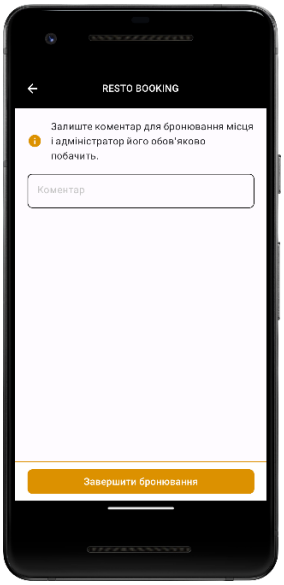



Рисунок 4.10– Екран бронювання

Для тих випадків, якщо користувач хотів би дізнатись якусь додаткову інформацію про заклад, чи коли він має додаткові запитання, у нього є можливість це зробити за вказаним на екрані номером телефону заклад. Для зручного режиму телефонного виклику без переключання між екранами було реалізовано виклик системного додатку з введеним номером, який дає можливість одразу зробити дзвінок (див. рисунок 4.11).


Рисунок 4.11– Системний додаток «Телефон», який відкривається одразу з введеним номером телефону закладу

Якщо заклад має додане статистику з опитувань, то на Toolbar буде відображатись іконка статистики. Якщо вона доступна, то після того як користувач натисне на іконку статистики, він переходить на екран з трьома діаграмами. Перші дві діаграми, які бачить користувач, це кругові діаграми, на яких видно який відсоток користувачів відповів «Так», або «Ні» на те, чи інше запитання. Відсоток користувачів які відповіли на питання «Так» зображено зеленим кольором, відсоток відповідей «Ні» - червоним (див. рисунок 4.12).


Рисунок 4.12 – Кругові діаграми на екрані «Статистика»

На даному екрані присутня ще гістограма, на якій зображено кількість користувачів, які обрали одну з п’яти даних відповідей, таких як: кухня, обслуговування, атмосфера, розташування, або нічого (див. рисунок 4.13).

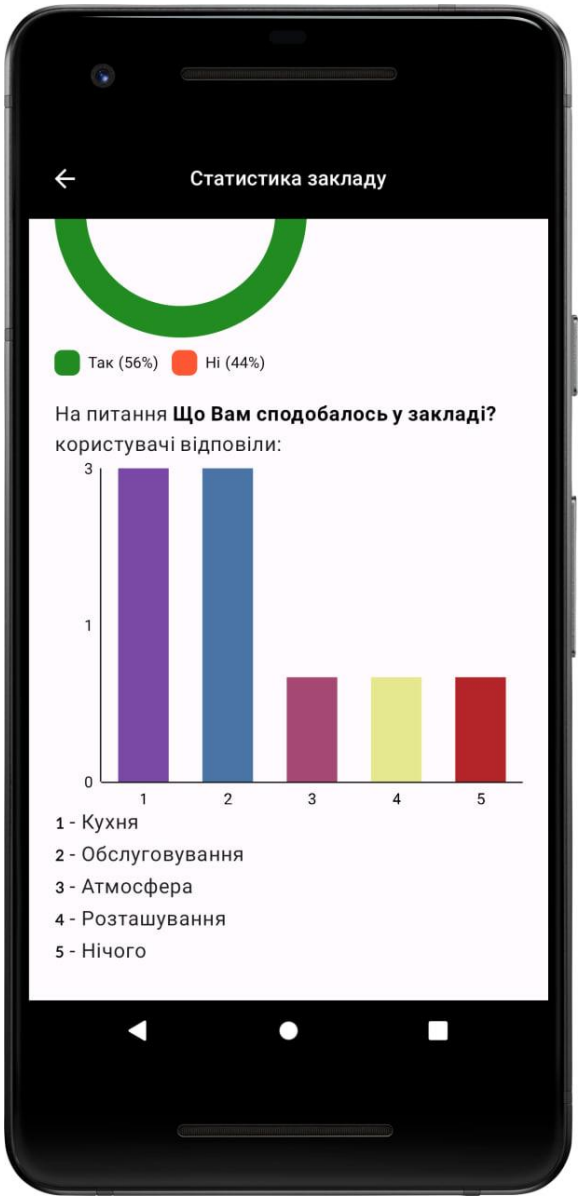



Рисунок 4.13 – Гістограма на екрані «Статистика»

Якщо повернутися на головний екран додатку і через нижній NavigationToolBar перейти на екран профіль, користувач зможе вийти з додатку, для того, щоб за потреби мати можливість увійти в систему під іншим акаунтом. Якщо у закладів, які зареєстровані з акаунта під яким авторизований користувач, є коментарі які не отримали відповідь, він побачить червоний «Label» з кількістю коментарів, які потребують відповіді. Екран «Профіль» зображено на рисунку 4.14.

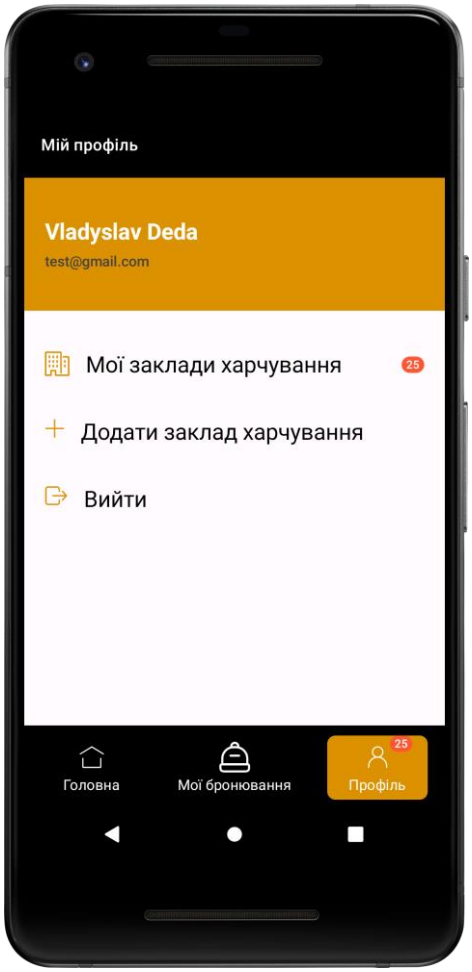



Рисунок 4.14 – Екран «Профіль»

На екрані «Профіль» напроти пункту «Мої заклади харчування» користувач теж побачить цю «Label» з кількістю коментарів без відповіді, для кращого розуміння чому з’явився цей знак. Натиснувши на пункт «Мої заклади харчування», користувачеві відкривається екран зі списком закладів, які були додані в систему з авторизованого акаунта.

Елементи списку містять основну інформацію про заклад, щоб адміністратор чи власник змогли його легко ідентифікувати, якщо він не один. Також, крім назви закладу, його рейтингу і адреси, елементи списку містять повідомлення, яке сповіщає, що заклад має коментарі без зворотного зв’язку, або ж навпаки все добре. Якщо заклад має коментарі, на які відповідь не була надана, то такий заклад буде обведено червоним кольором і сформується повідомлення червоним кольором, у якому вказано кількість коментарів для закладу без відповіді. Якщо ж у закладу немає таких коментарів, то у такого

закладу буде повідомлення зеленого кольору і буде сповіщатися, що усі клієнти отримали відповідь. Екран зі списком закладів харчування авторизованого користувача зображено на рисунку 4.15.

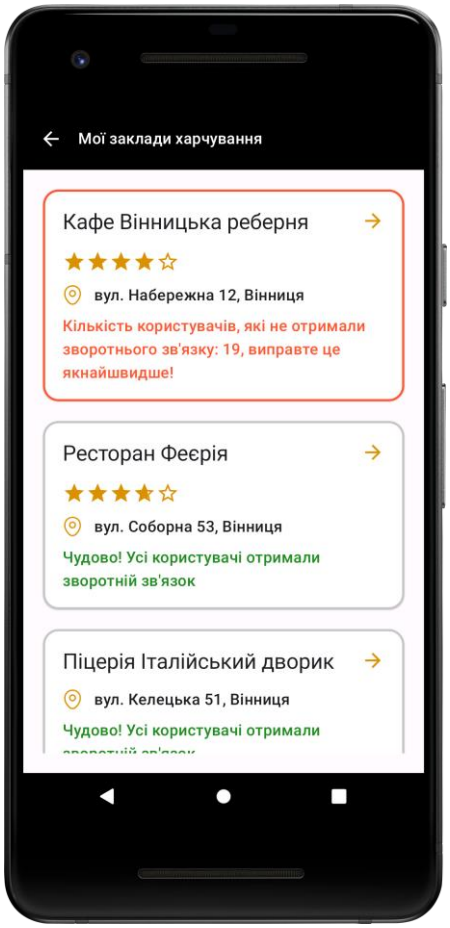



Рисунок 4.15 – Екран «Мої заклади харчування»

Натиснувши на елемент списку, користувач відкриє екран з деталями, які йому необхідні, а саме: заброньовані столики та усі коментарі. Якщо у закладу є коментарі, які залишились без зворотного зв’язку, то на екрані також відобразиться секція «Коментарі, які потребують відповіді» на червоному фоні. Натиснувши на останню секцію, користувачеві відкриється список коментарів без відповіді. У кожного коментаря є кнопка «Відповісти на коментар», після натискання на яку відкриється діалогове вікно з полем для введення коментаря і кнопкою «Відповісти». Екран, на якому зображено заклад, який має коментарі без відповіді, зображено на рисунку 4.16.

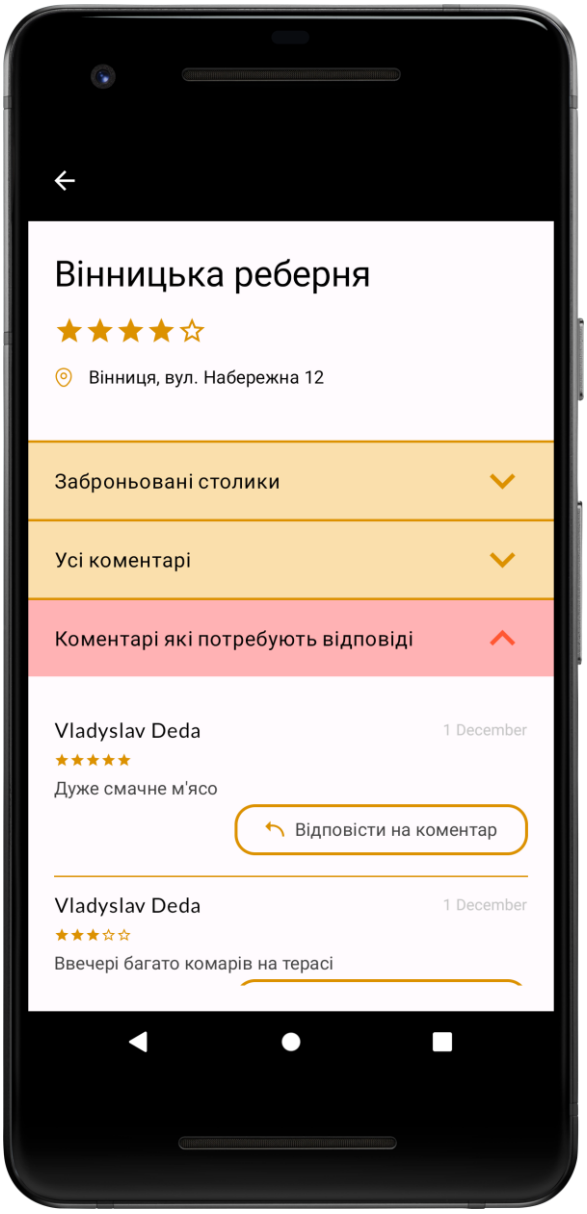



Рисунок 4.16 – Екран з деталями про заклад для користувача який зареєстрував цей заклад

Також на цьому екрані є можливість переглянути поточні бронювання для закладу харчування. Натиснувши на секцію «Заброньовані столики», користувачеві стає доступний список з забронюваними місцями у закладі харчування. Елементи списку містять інформацію про бронювання, а саме: дата та час бронювання, прізвище та ім’я того хто забронював місце, кількість столиків, а також замітки які залишив користувач з побажаннями чи коментарем. Екран з деталями про заклад для користувача, який зареєстрував цей заклад, з відкритим списком бронювань зображено на рисунку 4.17.

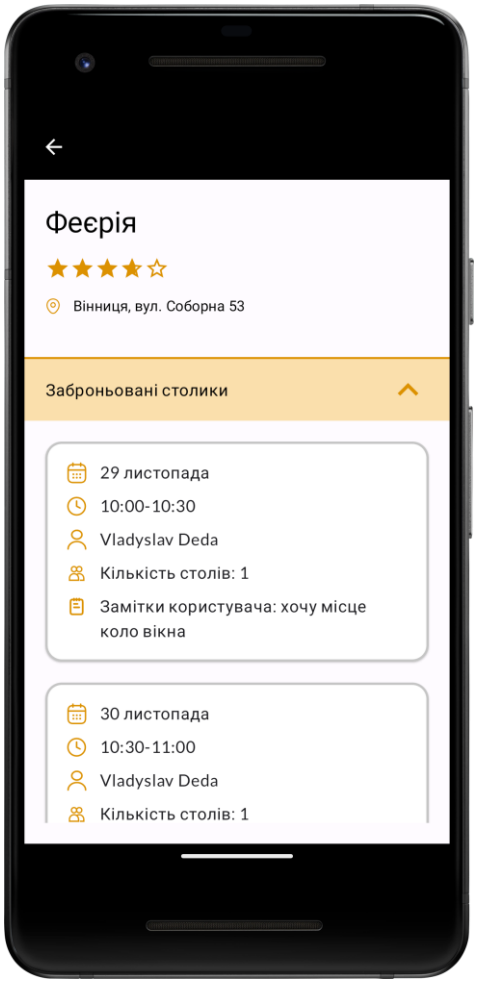



Рисунок 4.17 – Екран з деталями про заклад для користувача, який зареєстрував цей заклад з відкритою секцією «Заброньовані столики»

Також на цьому екрані реалізовано можливість додавання нового закладу харчування в систему. Для цього необхідно натиснути кнопку «Додати заклад харчування». Перший екран, який відкривається користувачеві, після того, як він натиснув на кнопку, це екран, де потрібно заповнити головну інформацію про заклад. На початку заповнюється назва закладу, місто, а потім адреса закладу, який реєструють. Також, необхідно ввести номер телефону, на який користувачі зможуть зателефонувати у разі виникнення додаткових запитань. Наступним кроком необхідно описати заклад, який додається. Далі зі списку меню потрібно обрати тип закладу, який додається: бар, ресторан, паб, тощо. Також потрібно вказати час початку роботи закладу та час, коли заклад зачиняється. Екран додавання головної інформації наведено на рисунку 4.18.

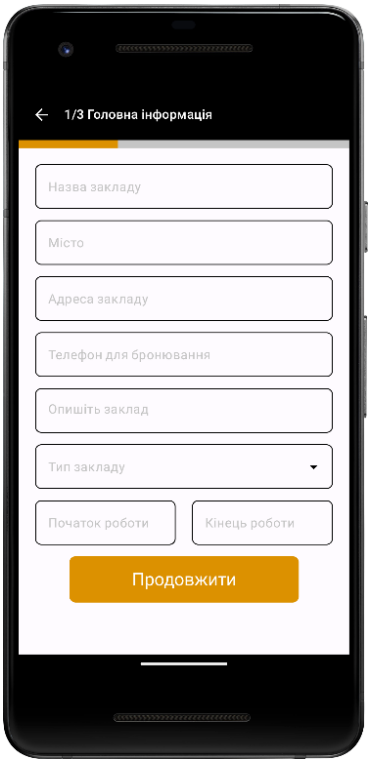



Рисунок 4.18 – Екран «Головна інформація» при додаванні нового закладу харчування

Після того, як користувач натисне меню часу початку чи закінчення роботи закладу, відкриється діалогове вікно, зображене на рисунку 4.19.


Рисунок 4.19 – Діалогове вікно додавання часу

Коли користувач натисне кнопку «Продовжити», він перейде на наступний екран додавання тегів. На цьому екрані запропоновано для вибору певну кількість тегів, які додані попередньо в системі. Також продемонстровано повідомлення, для чого необхідне додавання тегів та чому це важливо. Екран додавання тегів закладу харчування зображено на рисунку 4.20.

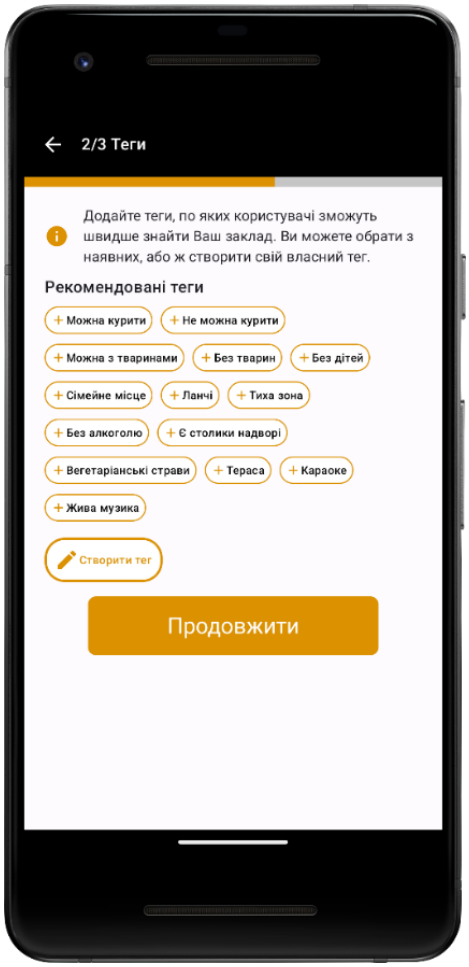



Рисунок 4.20 – Екран додавання тегів

Також у користувача є можливість створити власний тег для свого закладу, за яким користувачі зможуть швидко знайти саме цей заклад. Для того, аби створити власний тег, необхідно натиснути кнопку «Створити тег».

Після цього відкриється діалогове вікно, у якого є поле для вводу тексту тегу, а також кнопка «Додати», яка стає активною, після того, як поле введення тексту не буде порожнім. Діалогове вікно додавання тегів зображено на рисунку4.21.

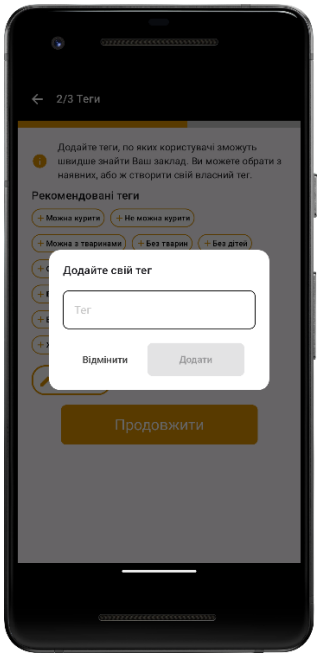



Рисунок 4.21 – Діалогове вікно додавання тегів

Після того, як користувач додасть теги і натисне кнопку «Продовжити», відбувається перехід на екран додавання фото. Щоб зареєструвати заклад, необхідно додати від однієї до шести фотографій закладу. Екран додавання фотографій закладу харчування зображено на рисунку 4.22.

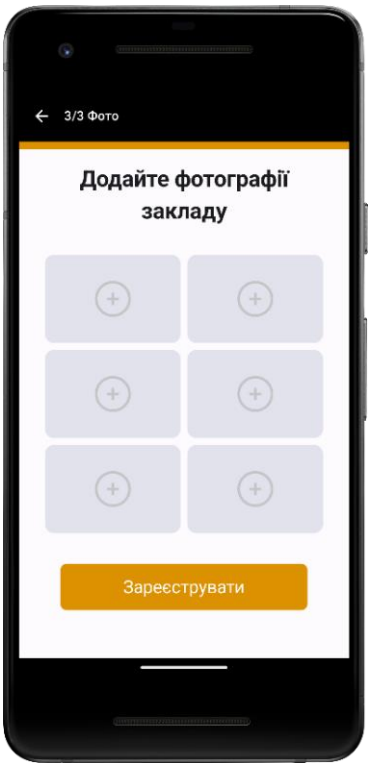



Рисунок 4.22 – Екран додавання фотографій закладу харчування

Після того, як користувач додасть всі фотографії, для завершення реєстрації потрібно натиснути кнопку «Зареєструвати». Після цього заклад додається в систему.

### 4.3 Розробка інструкції користувача

Інструкція для користувача передбачає ознайомлення з мінімальними технічними характеристиками для встановлення застосунку на смартфон. Детальний опис характеристик наведений у таблиці 4.1.

Таблиця 4.1 – Мінімальні технічні характеристики

| Характеристика | Опис |
|---|---|
| Операційна система | Android 8.0 або вище |
| Процесор | Мінімум 1,2 ГГц, чотирьохядерний або вище |
| Оперативна пам'ять | Мінімум 1 ГБ RAM |
| Вільне місце | Мінімум 40 МБ |
| Інтерфейси | Wi-Fi, LTE |

Для запуску мобільного додатку «Restobooking» необхідно запустити файл Restobooking, зображення іконки в головному меню має вигляд, як наведено на рисунку 4.23.


Рисунок 4.23 – Зображення іконки файлу програми в головному меню

Після того, як користувач успішно зареєструється чи авторизується, він переходить на головний екран програми. Після введеня міста та вибору хоча б одного тегу, користувач переходить на екран з відображенням результатів

пошуку. Для того, аби переглянути детальну інформацію про заклад, необхідно натиснути на заклад. Для того, щоб додати коментар закладу, необхідно натиснути на кнопку «+Додати коментар», після чого відкриється діалогове вікно з полем вводу коментаря та можливістю виставити оцінку та оцінити якість обслуговування. Також на екрані деталей закладу є можливість забронювати місце, натиснувши кнопку «Забронювати» і вказавши потрібні параметри у діалоговому вікні, яке відкриється. Перейшовши у вкладку «Профіль», у користувача є можливість додати новий заклад харчування в систему, вказавши всі дані на екранах, також є можливість вийти з поточного акаунту для можливості авторизації під іншим акаунтом.

### 4.4 Висновки

У четвертому розділі було проведено тестування програмних модулів мобільної Android-системи для бронювання місць і моніторингу якості обслуговування у закладах харчування. Були перевірені екрани з авторизації та реєстрації нового користувача, пошуку закладів, перегляд детальної інформації про заклад, додавання коментарів та додавання нових закладів у систему.

Розроблено інструкцію користувача з мінімальними характеристиками телефону для встановлення застосунку, а також інструкцію користувача, яка допоможе розпочати використання додатку та швидше ознайомитися зі способами використання функціоналу даного додатку.

Було описано мінімальні технічні характеристики для мобільного пристрою, які забезпечують стабільну та безперебійну роботу додатку.

# 5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проєкту, і переконати його в економічній доцільності такого кроку.

Для наведеного випадку мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

## 5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» є оцінювання

науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [32].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

| Бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено працездатність продукту в реальних умовах |
| Ринкові переваги (недоліки) | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в аналогів | Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів | Технічні та споживчі властивості продукту значно кращі, ніж в аналогів |
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |

Продовження табл. 5.1

| Бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| Практична здійсненність | | | | | |
| 8 | Відсутні фахівці як з технічної, так і з комерційної | Необхідно наймати фахівців або витрачати | Необхідне незначне навчання фахівців та | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела | Потрібні незначні фінансові ресурси. Джерела фінансування | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовують ся у військово | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуютьс |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності | Термін реалізації ідеї менше 3-х років. Термін окупності | Термін реалізації ідеї менше 3-х років. Термін окупності |
| 12 | Необхідна розробка регламентних документів та | Необхідно отримання великої кількості дозвільних | Процедура отримання дозвільних документів для | Необхідно тільки повідомлення відповідним органам про | Відсутні будь- які регламентні обмеження на виробництво та |

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

| Критерії | Експерт (ПІБ, посада) | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| | Бали: | | |
| 1. Технічна здійсненність концепції | 5 | 4 | 4 |
| 2. Ринкові переваги (наявність аналогів) | 4 | 4 | 4 |
| 3. Ринкові переваги (ціна продукту) | 4 | 3 | 3 |
| 4. Ринкові переваги (технічні властивості) | 4 | 4 | 3 |
| 5. Ринкові переваги (експлуатаційні витрати) | 3 | 3 | 3 |

Продовження табл. 5.2

| Критерії | Експерт (ПІБ, посада) | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| | Бали: | | |
| 6. Ринкові перспективи (розмір ринку) | 3 | 4 | 3 |
| 7. Ринкові перспективи (конкуренція) | 3 | 3 | 4 |
| 8. Практична здійсненність (наявність фахівців) | 3 | 3 | 3 |
| 9. Практична здійсненність (наявність фінансів) | 3 | 3 | 4 |
| 10. Практична здійсненність (необхідність нових матеріалів) | 3 | 3 | 3 |
| 11. Практична здійсненність (термін реалізації) | 3 | 3 | 4 |
| 12. Практична здійсненність (розробка документів) | 3 | 3 | 3 |
| Сума балів | 41 | 40 | 41 |
| Середньоарифметична сума балів $СБ_c$ | 40,7 | | |

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [33].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

| Середньоарифметична сума балів СБ, розрахована на основі висновків експертів | Науково-технічний рівень та комерційний потенціал розробки |
|---|---|
| 41…48 | Високий |
| 31…40 | Вище середнього |
| 21…30 | Середній |
| 11…20 | Нижче середнього |
| 0…10 | Низький |

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» становить 40,7 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

## 5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

### 5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою 5.1 [33]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де $k$ – кількість посад дослідників залучених до процесу досліджень;

$M_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

$t_i$ – число днів роботи конкретного дослідника, дні;

$T_p$ – середнє число робочих днів в місяці, $T_p$=22 дні.

$$З_о = 20000{,}00 \cdot 60 / 22 = 54545{,}45 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

| Найменування посади | Місячний посадовий оклад, грн | Оплата за робочий день, грн | Число днів роботи | Витрати на заробітну плату, грн |
|---|---|---|---|---|
| Керівник проекту | 20000 | 909,09 | 60 | 54545,45 |
| Інженер-розробник програмного забезпечення | 15000 | 681,82 | 60 | 40909,09 |
| Консультант | 15000 | 681,82 | 30 | 20454,55 |
| Всього | | | | 115909,01 |

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою 5.2:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (5.2)$$

де $C_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

$t_i$ – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду $C_i$ можна визначити за формулою 5.3:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (5.3)$$

де $M_M$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

$K_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [32];

$K_c$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_p$ – середнє число робочих днів в місяці, приблизно $T_p$ = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 69{,}09$ грн.

$З_{р1} = 69{,}09 \cdot 8{,}00 = 552{,}75$ грн.

Величина витрат на основну заробітну плату робітників наведена в табл.5.5.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

| Найменування робіт | Тривалість роботи, год | Розряд роботи | Тарифний коефіцієнт | Погодинна тарифна ставка, грн | Величина оплати на робітника грн |
|---|---|---|---|---|---|
| Підготовка робочого місця дослідника | 8 | 2 | 1,1 | 69,09 | 552,75 |
| Інсталяція програмного забезпечення | 3 | 5 | 1,7 | 106,78 | 320,34 |
| Всього | | | | | 873,09 |

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою 5.4:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (115909,09 + 873,09) · 11 / 100% = 12846,04 грн.

### 5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.5:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

*Зн* = (115909,09 + 873,09+ 12846,04) · 22 / 100% = 28518,21 грн.

### 5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.6:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

*n* – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 2 · 210,00 · 1,1 - 0,000 · 0,00 = 462,0 грн.

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

| Найменування матеріалу, марка, тип, сорт | Ціна за од | Норма витрат, од | Величина відходів, кг | Ціна відходів, грн/кг | Вартість витраченого матеріалу, грн |
|---|---|---|---|---|---|
| Офісний папір Calipso Plus A4-500-80 | 210 | 2 | 0 | 0 | 462 |
| Папір для записів Calipso Parers Light A5 | 110 | 1 | 0 | 0 | 121 |
| Органайзер офісний Calipso Office | 210 | 1 | 0 | 0 | 231 |
| Канцелярське приладдя (набір офісного працівника) | 175 | 3 | 0 | 0 | 577,5 |
| Картридж для принтера Canon LBP6500 | 1100 | 1 | 0 | 0 | 1210 |
| Flesh-пам'ять Kingston 32 GB | 180 | 1 | 0 | 0 | 198 |
| Тека для паперів CALIPSO BOX | 82 | 1 | 0 | 0 | 90,2 |
| | | | | | 2889,7 |

### 5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» відсутні.

### 5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного

для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Витрати на спецустаткування, які використовують при проведенні НДР на тему «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» відсутні.

### 5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою5.8:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$K_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

$k$ – кількість найменувань програмних засобів.

$B_{прг} = 11600,00 \cdot 4 \cdot 1,1 = 51968$ грн.

Отримані результати зведемо до таблиці 5.7.

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

| Найменування програмного засобу | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| ОС Windows 11 | 4 | 11600 | 51968 |

Продовження табл. 5.7

| Найменування програмного засобу | Кількість, шт | Ціна за одиницю, грн | Вартість, грн |
|---|---|---|---|
| Прикладний пакет Microsoft Office 2019 | 4 | 5500 | 24640 |
| Всього | | | 76608 |

### 5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 5.9:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (100000{,}00 \cdot 3) / (3 \cdot 12) =$ 8333,33 *грн.*

Проведені розрахунки зведемо до таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

| Найменування обладнання | Балансова вартість, грн | Строк корисного використання, років | Термін використання обладнання, місяців | Амортизаційні відрахування, грн |
|---|---|---|---|---|
| Персональний комп’ютер 4 шт | 100 000 | 3 | 3 | 8333,33 |

Продовження табл. 5.8

| Найменування обладнання | Балансова вартість, грн | Строк корисного використання, років | Термін використання обладнання, місяців | Амортизаційні відрахування, грн |
|---|---|---|---|---|
| Робоче місце дослідника | 15000 | 5 | 3 | 750,00 |
| Оргтехніка | 6000 | 4 | 3 | 375,00 |
| ОС Windows 11 (4 пак) | 51968 | 2 | 3 | 6496,00 |
| Прикладний пакет Microsoft Office 2019 (4 пак) | 24640 | 2 | 3 | 3080,00 |
| Всього | | | | 19034,33 |

### 5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_e$) розраховуємо за формулою 5.10:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.10)$$

де $W_{yi}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт;

$t_i$ – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

$\eta_i$ – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_е = 0,2 \cdot 480,0 \cdot 7,50 \cdot 0,95 / 0,97 = 2820,62$ *грн.*

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.9 – Витрати на електроенергію

| Найменування обладнання | Встановлена потужність, кВт | Тривалість роботи, год | Сума, грн |
|---|---|---|---|
| Персональний компютер (4 шт) | 0,2 | 480 | 2820,62 |
| Робоче місце дослідника | 0,15 | 480 | 528,87 |
| Оргтехніка | 0,45 | 20 | 66,11 |
| Всього | | | 3415,59 |

### 5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.11:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (5.11)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

$B_{св} = (263636,36 + 873,09) \cdot 20 / 100\% = 23356,44$ *грн.*

### 5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.12:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

*$B_{сп}$ = (115909,09 + 873,09) · 35 / 100% = 40873,76грн.*

### 5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.13:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

*$І_в$ = (115909,09 + 873,09) · 50 / 100% = 58391,09грн.*

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за

формулою 5.14:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

*$В_{нзв}$ = (115909,09 + 873,09) · 100 / 120% = 140 138,62 грн.*

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою 5.15:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

*$В_{заг}$=115909,09+873,09+12846,04+28518,21+2889,7+0,00+76608+19034,33+ 3415,59+ 23356,44+40873,76+58391,09+140 138,62= 522853,98грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 5.16:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.16)$$

де $\eta$ - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо $\eta$=0,7.

*ЗВ = 522853,98/ 0,7 = 746934,25 грн.*

## 5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$\Delta N$ – збільшення кількості споживачів продукту, у періоди часу, що

аналізуються, від покращення його певних характеристик;

1-й рік – 5000 користувачів/рік;

2-й рік – 6000 користувачів/рік;

3-й рік – 3500 користувачів/рік.

$N$ – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 100000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 11100,00 грн /за рік користування;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою 5.17 [31]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.17)$$

де $\lambda$ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

$\rho$ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho$ =30%;

$\vartheta$ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (100000{,}00 \cdot 500{,}00 + 11600 \cdot 15000) \cdot 0{,}83 \cdot 0{,}3 \cdot (1\text{-}0{,}18/100\%) =$ 138114732 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(100000,00·500,00+11600·(15000+20000))·0,83·0,3·(1-0,18/100%)= 185655708 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(100000,00·500,00+11600·(15000+20000+13500))·0,83·0,3·(1-0,18/100%)= 217745866,8 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою 5.18:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

$T$ – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau$ =0,25;

$t$ – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

*ПП* = 138114732/(1+0,25)$^1$+ 185655708/(1+0,25)$^2$+ 217745866,8/(1+0,25)$^3$= 340797322,52 грн.

Величина початкових інвестицій $PV$, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки, розраховується за формулою 5.19:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=4;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 746934,25 грн.

$PV = k_{інв} \cdot ЗВ$ = 4· 746934,25 = 2987737,006 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки розраховується за формулою 5.20:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 340797322,52 грн;

$PV$ – теперішня вартість початкових інвестицій, 2987737,006 грн.

$E_{абс} = ПП - PV$ = 340797322,52 - 2987737,006 = 337809585,52грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.21:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 337809585,52грн;

$PV$ – теперішня вартість початкових інвестицій, 2987737,006 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+337809585{,}52/2987737{,}006)^{1/3} = 3{,}85.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ розраховується за формулою 5.22:

$$\tau_{мін} = d + f, \tag{5.22}$$

де $d$ – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні $d$ =0,11;

$f$ – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 3,85 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 5.23:

$$T_{ок} = \frac{1}{E_в}, \tag{5.23}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 3,85 = 0,26 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

## 5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» становить 40,7 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,26 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування».

# ВИСНОВКИ

У під час виконання магістерської кваліфікаційної роботи було розроблено методи та програмні засоби Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування. Робота оформлена згідно методичних вказівок [34, 35].

Було проаналізовано стан даного питання на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і зроблено порівняння з власним програмним продуктом. У результаті аналізу обрано мову програмування Kotlin, інтегроване середовище розробки Android Studio. Вибрано базу даних Cloud Firestore для зберігання даних користувачів. Для зберігання фото використано Firebase Storage, а для авторизації – Firebase Auth.

Розроблено Android-систему «RestoBooking» для бронювання місць і моніторингу якості обслуговування у закладах харчування, яка включає модуль реєстрації користувача, модуль реєстрації нового закладу харчування, модуль додавання коментарів, модуль завантаження усіх закладів харчування, модуль завантаження закладу харчування за ID, модуль бронювання місць у закладах.

Розроблено метод пошуку закладів харчування для бронювання місць, який, має персоналізований підхід до пошуку та застосовує адаптивні алгоритми з індивідуалізацією пошукових процесів, що дозволяє покращити інтерактивні комунікації з клієнтами;

Розроблено метод моніторингу якості обслуговування в закладах харчування, який реалізує в середовищі мобільної системи спеціалізовані засоби підтримки інтерактивної взаємодії працівників з клієнтами, що впливає на визначення статусу закладу харчування в системі рейтингового оцінювання в процесі формування списку пріоритетних варіантів і сприяє удосконаленню бізнес-стратегії закладів харчування.

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

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.
2. Quand les nouvelles technologies s'invitent à notre table Electronic resource. [Електронний ресурс] – Режим доступу до ресурсу: https://www.restoconnection.fr/commentutiliser-les-nouvelles- technologies-dans-les-restaurants/
3. Войтко В.В. Особливості розробки мобільного Android-додатку «RESTOBOOKING» для моніторингу якості обслуговування в закладах харчування / В.В. Войтко, Н.Є. Барчук, О.В. Гаврилюк, В.П. Деда // Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. – Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – С. 64-68.
4. Огляд на Android 13 зі сторони зручності використання операційної системи [Електронний ресурс] – Режим доступу до ресурсу: https://cases.media/article/oglyad-na-android-13-zi-storoni-zruchnosti-vikoristannya-operaciinoyi-sistemi
5. Comment [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/org/w3c/dom/Comment
6. TheTable [Електронний ресурс] – Режим доступу до ресурсу: https://thetable.app.
7. Сервіс для бронювання столиків Reservble тепер працює в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://nashkiev.ua/news/servis-dlya-bronyuvannya-stolikiv-reservble-teper-pratsyue-v-ukraini.
8. Кращі заклади Києва. Резерв столика зі знижкою або подарунком в кращих закладах [Електронний ресурс] – Режим доступу до ресурсу: https://reston.ua.

9. timeat [Електронний ресурс] – Режим доступу до ресурсу: https://jobs.dou.ua/companies/timeat/.
10. Система лояльності для ресторанів, до яких гості захочуть повернутися [Електронний ресурс] – Режим доступу до ресурсу: https://takeuseat.in.ua.
11. Кросплатформова розробка додатків [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/krossplatformennaya-razrabotka-prilozhenij.
12. Переваги та недоліки кросплатформної та нативної розробки мобільних додатків [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ua/blog/cross-platform-native-mobile-development/.
13. Layouts in Views [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/develop/ui/views/layout/declaring-layout.
14. State and Jetpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose/state.
15. Jetpack Compose phases [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose/phases
16. Material Design: The Complete Guide / Nick Babich // Smashing Magazine – 2015. – C. 32-112.
17. Compose modifiers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose/modifiers.
18. Preview UI in Compose [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose/tooling/previews.
19. State and Jetpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose/state.
20. Remember in jetpack compose [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@sujathamudadla1213/remember-in-jetpack-compose-720537356786.
21. Getting Started With MVVM in Jetpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://betterprogramming.pub/mvvm-in-jetpack-compose-part-4-fe757a1a1b84.
22. Build better apps faster with Jetpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose.

23. Data classes [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/docs/data-classes.html.
24. Android Studio: переваги та особливості [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/.
25. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.
26. Діаграма прецедентів [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikiwand.com/uk/Діаграма_прецедентів.
27. What exactly is Kotlin? [Електронний ресурс] – Режим доступу до ресурсу: https://codeop.tech/what-exactly-is-kotlin/.
28. Introduction to Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@rohinideshmane.21/introduction-to-kotlin.
29. Wim J. van der Linden, Cees A.W. Glas. Computerized Adaptive Testing: Theory and Practice. – Dordrecht, The Netherlands: Kluwer, 2000. – 323 p.
30. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017.
31. Що таке тестування програмного забезпечення? [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya/.
32. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.
33. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.
34. Положення про кваліфікаційну роботу на другому (вищому) рівні вищої освіти. Розробники: А.О. Семенов, Л.П. Громова, Т.В. Макарова, О.В. Сердюк. Вінниця : ВНТУ, 2021. – 60 с.

35. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.

# ДОДАТКИ

# Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
к.т.н., проф. О. Н. Романюк
"19" вересня 2023 р.

**Технічне завдання**
**на магістерську кваліфікаційну роботу «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування» за спеціальністю**
**121 – Інженерія програмного забезпечення**

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ. В.В. Войтко
"19" вересня 2023 р.

Виконав:
студент гр.3ПІ-22м В. П. Деда
"19" вересня 2023 р.

Вінниця – 2023 року

## 1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування».Галузь застосування – мобільні системи.

## 2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора № 247 від 18 вересня 2023 р. по ВНТУ про закріплення тем МКР.

## 3. Мета та призначення розробки.

Метою роботи є підвищенням ефективності пошуку і бронювання місць у закладах харчування шляхом розробки спеціалізованого Android застосунку з удосконаленням методів і засобів пошуку закладів, що дозволить пришвидшити пошукові процеси та сприятиме розвитку комунікацій з клієнтами в інтерактивному режимі.

Призначення роботи – розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування

## 4. Вихідні дані для проведення НДР

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

1. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.
2. Kotlin coroutines on Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/kotlin/coroutines.
3. Романюк О.Н. Організація баз даних і знань. / О.Н. Романюк, Т.О. Савчук //Навчальний посібник. – Вінниця: УНІВЕРСУМ –Вінниця. –2003. – 123 с.

**5. Технічні вимоги**

Вхідні дані до роботи – авторизований користувач, теги, заклади харчування, бронювання, коментарі, опитування; вихідні дані – графічний інтерфейс з можливістю бронювання місць і моніторингу якості обслуговування в закладах харчування.

**6. Конструктивні вимоги.**

Інтерфейс програми повинен відповідати естетичним та ергономічним вимогам та мати зручну навігацію і засоби керування.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

**9. Стадії та етапи розробки:**

| № з/п | Назва етапів магістерської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1 | Аналіз стану розвитку мобільних систем для бронювання місць та постановка задач дослідження | 20.09.2023<br>30.09.2023 |
| 2 | Розробка методів та моделей Android-системи | 01.10.2023<br>17.10.2023 |
| 3 | Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування | 18.10.2023<br>04.11.2023 |
| 4 | Тестування програми | 05.11.2023<br>21.11.2023 |
| 5 | Економічна частина | 22.11.2023<br>01.12.2023 |

**10. Порядок контролю та прийняття.**

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

## Додаток Б – Протокол перевірки на плагіат

## ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів та програмних засобів Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник: к.т.н., доц. каф. ПЗ. В.В. Войтко

| Unicheck | |
|---|---|
| **Оригінальність** | **87,1 %** |
| Схожість | 12,9 % |

### Аналіз звіту подібності

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

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Деда В.П.

Керівник роботи _____ Войтко В.В.

# Додаток В – Лістинг програми

## AppNavigation.kt

```
package com.project.presentation.ui.navigation

import androidx.annotation.DrawableRes
import androidx.compose.animation.core.tween
import androidx.compose.animation.fadeIn
import androidx.compose.animation.fadeOut
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.navArgument
import com.project.presentation.R
import com.project.presentation.ui.navigation.SrpDestinationsArgs.CITY_ARG
import com.project.presentation.ui.navigation.SrpDestinationsArgs.TAGS_ARG
import
com.project.presentation.ui.screens.add_food_establishments.AddFoodEstablishmentsScreen
import com.project.presentation.ui.screens.home.HomeScreen
import com.project.presentation.ui.screens.login.LoginScreen
import com.project.presentation.ui.screens.profile.ProfileScreen
import com.project.presentation.ui.screens.signup.SignUpScreen
import com.project.presentation.ui.screens.spalsh.SplashScreen
import com.project.presentation.ui.screens.srp.SrpScreen

object SrpDestinationsArgs {
    const val CITY_ARG = "city"
    const val TAGS_ARG = "tags"
}

sealed class AppDestinations(
    val route: String,
    val title: String? = null,
    @DrawableRes
    val icon: Int? = null
) {
    object Splash : AppDestinations("splash_screen")
    object Login : AppDestinations("login_screen")
    object SignUp : AppDestinations("sign_up_screen")
    object Home : AppDestinations("home_screen", "Головна", R.drawable.ic_home)
    object Reservations :
        AppDestinations("reservations_screen", "Мої заклади",
R.drawable.ic_my_reservation)

    object Profile : AppDestinations("profile_screen", "Профіль", R.drawable.ic_profile)
    object AddFoodEstablishments : AppDestinations("add_food_establishments")
    object Srp : AppDestinations("srp_screen")
}

@Composable
fun SetupNavGraph(
    modifier: Modifier = Modifier,
    navController: NavHostController
) {
    NavHost(
        navController = navController,
        startDestination = AppDestinations.Splash.route,
        modifier = modifier,
    ) {
```

```
composable(
    route = AppDestinations.Splash.route,
    enterTransition = {
        fadeIn(
            animationSpec = tween(700)
        )
    },
    exitTransition = {
        fadeOut(
            animationSpec = tween(700)
        )
    }
) {
    SplashScreen(
        navigateLogin = {
            navController.navigate(AppDestinations.Login.route) {
                launchSingleTop = true
                popUpTo(0) { inclusive = true }
            }
        },
        navigateHome = {
            navController.navigate(AppDestinations.Home.route) {
                launchSingleTop = true
                popUpTo(0) { inclusive = true }
            }
        }
    )
}

composable(
    route = AppDestinations.Login.route,
    enterTransition = {
        fadeIn(
            animationSpec = tween(700)
        )
    },
    exitTransition = {
        fadeOut(
            animationSpec = tween(700)
        )
    }) {
    LoginScreen(
        homeNavigate = {
            navController.navigate(AppDestinations.Home.route) {
                launchSingleTop = true
                popUpTo(0) { inclusive = true }
            }
        },
        signUpNavigate = {
            navController.navigate(AppDestinations.SignUp.route)
        }
    )
}

composable(
    route = AppDestinations.SignUp.route,
    enterTransition = {
        fadeIn(
            animationSpec = tween(700)
        )
    },
    exitTransition = {
        fadeOut(
            animationSpec = tween(700)
        )
    }) {
    SignUpScreen(
        homeNavigate = {
            navController.navigate(AppDestinations.Home.route) {
                launchSingleTop = true
                popUpTo(0) { inclusive = true }
```

```
                    }
                }
            )
        }
        composable(
            route = AppDestinations.Home.route,
            enterTransition = {
                fadeIn(
                    animationSpec = tween(700)
                )
            },
            exitTransition = {
                fadeOut(
                    animationSpec = tween(700)
                )
            }) {
            HomeScreen(
                navigateToSrp = { city, tags ->
                    navController.navigate("${AppDestinations.Srp.route}/$city/$tags")
                }
            )
        }
        composable(
            route = AppDestinations.Reservations.route,
            enterTransition = {
                fadeIn(
                    animationSpec = tween(700)
                )
            },
            exitTransition = {
                fadeOut(
                    animationSpec = tween(700)
                )
            }) {
            Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
                Text(text = "Reservations Screen", style =
MaterialTheme.typography.bodyLarge)
            }
        }
        composable(
            route = AppDestinations.Profile.route,
            enterTransition = {
                fadeIn(
                    animationSpec = tween(700)
                )
            },
            exitTransition = {
                fadeOut(
                    animationSpec = tween(700)
                )
            }) {
            ProfileScreen(
                navigateToAddFoodEstablishment = {
                    navController.navigate(AppDestinations.AddFoodEstablishments.route)
                },
                navigateToLoginScreen = {
                    navController.navigate(AppDestinations.Login.route) {
                        launchSingleTop = true
                        popUpTo(0) { inclusive = true }
                    }
                }
            )
        }
        composable(
            route = AppDestinations.AddFoodEstablishments.route,
            enterTransition = {
                fadeIn(
                    animationSpec = tween(700)
                )
            },
            exitTransition = {
```

```
                fadeOut(
                    animationSpec = tween(700)
                )
            }) {
            AddFoodEstablishmentsScreen(
                navigateBack = {
                    navController.navigate(AppDestinations.Profile.route) {
                        launchSingleTop = true
                        popUpTo(0) { inclusive = true }
                    }
                },
                navigateToProfileScreen = {
                    navigateAndClearBackStack(AppDestinations.Profile.route,
navController)
                }
            )
        }
        composable(
            route = "${AppDestinations.Srp.route}/{$CITY_ARG}/{$TAGS_ARG}",
            arguments = listOf(
                navArgument(CITY_ARG) { type = NavType.StringType },
                navArgument(TAGS_ARG) { type = NavType.StringArrayType }),
            enterTransition = {
                fadeIn(
                    animationSpec = tween(700)
                )
            },
            exitTransition = {
                fadeOut(
                    animationSpec = tween(700)
                )
            }) {
            SrpScreen(
                navigateBack = {
                    navController.navigate(AppDestinations.Home.route) {
                        launchSingleTop = true
                        popUpTo(0) { inclusive = true }
                    }
                }
            )
        }
    }

}

fun navigateAndClearBackStack(route: String, navController: NavHostController) {
    navController.navigate(route) {
        launchSingleTop = true
        popUpTo(0) { inclusive = true }
    }
}
```

## FoodEstablishmentRepositoryImpl.kt

```
package com.project.data.repository

import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.firestore.ktx.toObject
import com.google.firebase.storage.StorageReference
import com.project.data.mapper.Mapper.toDomain
import com.project.data.mapper.Mapper.toDto
import com.project.data.model.FoodEstablishmentDto
import com.project.domain.model.FoodEstablishment
import com.project.domain.repository.FoodEstablishmentRepository
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.tasks.await
import kotlinx.coroutines.withContext
```

```
import javax.inject.Inject

private const val FOOD_ESTABLISHMENT_COLLECTION = "food_establishments"

class FoodEstablishmentRepositoryImpl @Inject constructor(
    private val storage: StorageReference,
    private val firestore: FirebaseFirestore
) : FoodEstablishmentRepository {


    override suspend fun registerFoodEstablishment(foodEstablishment:
FoodEstablishment): Result<Unit> =
        withContext(Dispatchers.IO) {
            try {
                val imagesUrls = mutableListOf<String>()
                foodEstablishment.photoList.forEach {
                    val ref =
storage.child("${foodEstablishment.name}_${it.index}")
                    it.uri?.let { it1 -> ref.putFile(it1).await() }
                    val url: String = ref.downloadUrl.await().toString()
                    imagesUrls.add(url)
                }
                firestore.collection(FOOD_ESTABLISHMENT_COLLECTION)
                    .add(foodEstablishment.toDto(imagesUrls)).await()
                Result.success(Unit)
            } catch (e: Exception) {
                Result.failure(e)
            }
        }

    override suspend fun fetchFoodEstablishments(
        city: String,
        tags: List<String>
    ): Result<List<FoodEstablishment>> =
        withContext(Dispatchers.IO) {
            try {
                val collection =
firestore.collection(FOOD_ESTABLISHMENT_COLLECTION)
                val query = if (city.isNotEmpty()) {
                    collection.whereEqualTo("city", city)
                } else {
                    collection
                }
                val task = query.get().await()
                val posts = task.documents.mapNotNull { documentSnapshot ->

documentSnapshot.toObject<FoodEstablishmentDto>()?.toDomain()
                }
                Result.success(posts)
            } catch (e: Exception) {
                Result.failure(e)
            }
        }
}
```

## UserRepositoryImpl.kt

```
package com.project.data.repository

import com.google.firebase.auth.FirebaseAuth
```

```
import com.google.firebase.auth.UserProfileChangeRequest
import com.project.domain.model.CreateUser
import com.project.domain.repository.UserRepository
import kotlinx.coroutines.tasks.await
import javax.inject.Inject

class UserRepositoryImpl @Inject constructor(
    private val auth: FirebaseAuth
) : UserRepository {

    override val currentUser get() = auth.currentUser

    override suspend fun firebaseSignInWithEmailAndPassword(
        email: String,
        password: String
    ): Result<Unit> =
        try {
            auth.signInWithEmailAndPassword(email, password).await()
            Result.success(Unit)
        } catch (e: Exception) {
            Result.failure(e)
        }


    override suspend fun firebaseSignUpNewUser(user: CreateUser): Result<Unit> =
        try {
            val result = auth.createUserWithEmailAndPassword(user.email,
user.password).await()
            val profileUpdates = UserProfileChangeRequest.Builder()
                .setDisplayName(user.nameSurname)
                .build()
            result.user?.updateProfile(profileUpdates)?.await()
            Result.success(Unit)
        } catch (e: Exception) {
            Result.failure(e)
        }


    override fun logout() = auth.signOut()

    override fun isUserLoggedIn() = currentUser != null
}
```

## Mapper.kt

```
package com.project.data.mapper

import android.annotation.SuppressLint
import android.net.Uri
import com.project.data.model.FoodEstablishmentDto
import com.project.domain.model.FoodEstablishment
import com.project.domain.model.Photo

@SuppressLint("NewApi")
object Mapper {

    fun FoodEstablishment.toDto(imageUrlList: List<String>):
FoodEstablishmentDto {
        return FoodEstablishmentDto(
            name = this.name,
```

```
            foodEstablishmentType = this.foodEstablishmentType,
            address = this.address,
            description = this.description,
            photoUrlList = imageUrlList,
            ownerName = this.ownerName,
            city = this.city,
            selectedTimeFrom = this.selectedTimeFrom,
            selectedTimeTo = this.selectedTimeTo,
            phoneForBooking = this.phoneForBooking,
            tags = this.tags
        )

    }

    fun FoodEstablishmentDto.toDomain(): FoodEstablishment {
        return FoodEstablishment(
            name = this.name,
            foodEstablishmentType = this.foodEstablishmentType,
            address = this.address,
            description = this.description,
            photoList = this.photoUrlList.mapIndexedNotNull { index, uri ->
                Photo(
                    index,
                    Uri.parse(uri)
                )
            },
            ownerName = this.ownerName,
            city = this.city,
            selectedTimeFrom = this.selectedTimeFrom,
            selectedTimeTo = this.selectedTimeTo,
            phoneForBooking = this.phoneForBooking,
            tags = this.tags
        )
    }
}
```

## HomeSearchView.kt

```
package com.project.presentation.ui.view

import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.ExperimentalLayoutApi
import androidx.compose.foundation.layout.FlowRow
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
```

```
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.Info
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.FilterChip
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Text
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalFocusManager
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import com.project.presentation.R
import com.project.presentation.ui.view.register_food_establishment.Tag
import com.project.presentation.ui.view.register_food_establishment.TagView
import java.time.LocalDate

@Composable
@OptIn(ExperimentalMaterial3Api::class, ExperimentalLayoutApi::class)
fun HomeSearchView(
    modifier: Modifier = Modifier,
    viewState: HomeSearchViewState,
    onCityChanged: (String) -> Unit,
    handleTagClick: (Tag) -> Unit,
    onSearchClicked: () -> Unit,
) {
    val focusManager = LocalFocusManager.current

    Column(
        modifier = Modifier
            .padding(20.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        OutlinedTextField(
            value = viewState.city,
            modifier = modifier
                .fillMaxWidth(),
            onValueChange = onCityChanged,
            placeholder = { Text(text = stringResource(id = R.string.city)) },
            leadingIcon = {
                Icon(
                    painterResource(id = R.drawable.ic_place),
                    contentDescription = null,
                    tint = colorResource(
                        id = R.color.main_yellow
                    )
                )
            },
            keyboardActions = KeyboardActions(onNext = {
focusManager.clearFocus() }),
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Text),
            colors = TextFieldDefaults.outlinedTextFieldColors(
```

```
                cursorColor = colorResource(id = R.color.gray),
                textColor = Color.Black,
                placeholderColor = colorResource(id = R.color.gray),
                focusedBorderColor = Color.Black,
                unfocusedBorderColor = Color.Black
            ),
            singleLine = true,
            shape = RoundedCornerShape(8.dp),
        )
        Spacer(modifier = Modifier.height(20.dp))
        Row(
            verticalAlignment = Alignment.CenterVertically
        ) {
            Icon(
                imageVector = Icons.Default.Info,
                tint = colorResource(id = R.color.main_yellow),
                contentDescription = null
            )
            Spacer(modifier = Modifier.width(14.dp))
            Text(
                text = stringResource(R.string.chose_tags),
                style = MaterialTheme.typography.bodyMedium
            )
        }
        Spacer(modifier = Modifier.height(8.dp))
        Box(
            modifier = Modifier
                .padding(
                    vertical = 2.dp,
                )
                .border(
                    width = 1.dp,
                    color = Color.Black,
                    shape = RoundedCornerShape(20.dp)
                )
                .background(
                    color = Color.White,
                    shape = RoundedCornerShape(20.dp)
                )
                .clip(shape = RoundedCornerShape(20.dp))
                .padding(10.dp)
                .align(Alignment.Start)
                .fillMaxSize()
                .weight(1f)
        ) {
            FlowRow(
                modifier = Modifier
                    .fillMaxWidth()
                    .verticalScroll(rememberScrollState()),
                horizontalArrangement = Arrangement.spacedBy(8.dp,
Alignment.Start),
                verticalArrangement = Arrangement.spacedBy(6.dp,
Alignment.CenterVertically),
            ) {
                viewState.tags.forEach { tag ->
                    TagView(
                        tag = tag,
                        onClicked = {
                            handleTagClick(tag)
                        }
                    )
                }
            }
        }
```

```
        Spacer(modifier = Modifier.height(20.dp))
        Button(
            onClick = onSearchClicked,
            shape = RoundedCornerShape(8.dp),
            enabled = viewState.isSearchButtonEnabled(),
            colors = ButtonDefaults.buttonColors(containerColor =
colorResource(id = R.color.main_yellow))
        ) {
            Text(
                modifier = Modifier.padding(vertical = 8.dp, horizontal =
54.dp),
                text = stringResource(id = R.string.search),
                style = MaterialTheme.typography.titleLarge
            )
        }
        Spacer(modifier = Modifier.height(80.dp))
    }
}
```

## EmailPasswordView.kt

```
package com.project.presentation.ui.view

import androidx.annotation.DrawableRes
import androidx.annotation.StringRes
import androidx.compose.animation.AnimatedVisibility
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.focus.onFocusChanged
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalFocusManager
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.project.presentation.R
```

```
@Composable
fun EmailPasswordView(
    modifier: Modifier = Modifier,
    viewState: EmailPasswordViewState,
    onEmailTextChange: (String) -> Unit,
    onPasswordTextChange: (String) -> Unit,
    onPasswordVisibilityClick: () -> Unit,
    onSignInClicked: () -> Unit
) {
    Column(
        modifier = modifier
            .fillMaxWidth(),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        val focusManager = LocalFocusManager.current
        TextInputField(
            value = viewState.email,
            onValueChange = onEmailTextChange,
            isError = viewState.errorMessage?.isNotEmpty() == true,
            labelText = stringResource(viewState.emailLabelText),
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Email)
        ) {
            focusManager.clearFocus()
        }
        Spacer(modifier = Modifier.height(20.dp))
        PasswordInputField(
            value = viewState.password.password,
            onValueChange = onPasswordTextChange,
            isError = viewState.errorMessage?.isNotEmpty() == true,
            labelText = viewState.password.labelText,
            showPassword = viewState.password.showPassword,
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Password),
            endIcon = viewState.password.endIcon(),
            clearFocus = { focusManager.clearFocus() },
            onPasswordVisibilityClick = onPasswordVisibilityClick
        )
        Spacer(modifier = Modifier.height(20.dp))
        AnimatedVisibility(viewState.errorMessage?.isNotEmpty() == true) {
            Column {
                Text(
                    modifier = Modifier.fillMaxWidth(),
                    text = viewState.errorMessage ?: "",
                    style = MaterialTheme.typography.titleSmall.copy(color =
MaterialTheme.colorScheme.error),
                    textAlign = TextAlign.Center
                )
                Spacer(modifier = Modifier.height(20.dp))
            }
        }
        Button(
            onClick = onSignInClicked,
//            enabled = viewState.isButtonEnabled(),
            shape = RoundedCornerShape(8.dp),
            colors = ButtonDefaults.buttonColors(containerColor =
colorResource(id = R.color.main_yellow))
        ) {
            Text(
                modifier = Modifier.padding(vertical = 8.dp, horizontal =
54.dp),
                text = stringResource(id = R.string.sign_in),
                style = MaterialTheme.typography.titleLarge
            )
        }
```

```
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TextInputField(
    modifier: Modifier = Modifier,
    value: String,
    onInputFieldFocusChanged: (Boolean) -> Unit = {},
    onValueChange: (String) -> Unit,
    isError: Boolean = false,
    labelText: String,
    keyboardOptions: KeyboardOptions,
    clearFocus: () -> Unit
) {
    TextField(
        value = value,
        modifier = modifier
            .fillMaxWidth()
            .onFocusChanged {
                onInputFieldFocusChanged(it.isFocused)
            },
        onValueChange = onValueChange,
        isError = isError,
        placeholder = { Text(text = labelText) },
        keyboardActions = KeyboardActions(onNext = { clearFocus() }),
        keyboardOptions = keyboardOptions,
        colors = TextFieldDefaults.textFieldColors(
            containerColor = Color.White,
            cursorColor = colorResource(id = R.color.gray),
            focusedIndicatorColor = Color.Transparent,
            unfocusedIndicatorColor = Color.Transparent,
            textColor = Color.Black,
            placeholderColor = colorResource(id = R.color.gray)
        ),
        shape = RoundedCornerShape(8.dp),
        singleLine = true,
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun PasswordInputField(
    modifier: Modifier = Modifier,
    value: String,
    onInputFieldFocusChanged: (Boolean) -> Unit = {},
    onValueChange: (String) -> Unit,
    isError: Boolean,
    @StringRes
    labelText: Int,
    showPassword: Boolean,
    keyboardOptions: KeyboardOptions,
    @DrawableRes
    endIcon: Int,
    clearFocus: () -> Unit,
    onPasswordVisibilityClick: () -> Unit
) {
    TextField(
        value = value,
        modifier = modifier
            .fillMaxWidth()
            .onFocusChanged {
                onInputFieldFocusChanged(it.isFocused)
            },
```

```
        onValueChange = onValueChange,
        isError = isError,
        singleLine = true,
        placeholder = { Text(text = stringResource(id = labelText)) },
        visualTransformation = if (showPassword) VisualTransformation.None else
PasswordVisualTransformation(),
        keyboardActions = KeyboardActions(onNext = { clearFocus() }),
        keyboardOptions = keyboardOptions,
        colors = TextFieldDefaults.textFieldColors(
            containerColor = Color.White,
            cursorColor = colorResource(id = R.color.gray),
            focusedIndicatorColor = Color.Transparent,
            unfocusedIndicatorColor = Color.Transparent,
            textColor = Color.Black,
            placeholderColor = colorResource(id = R.color.gray)
        ),
        shape = RoundedCornerShape(8.dp),
        trailingIcon = {
            IconButton(onClick = { onPasswordVisibilityClick() }) {
                Icon(
                    painterResource(id = endIcon),
                    contentDescription = "password visibility"
                )
            }
        },
    )
}

@Preview(
    showBackground = true
)
@Composable
fun EmailPasswordViewPreview() {
    EmailPasswordView(
        viewState = EmailPasswordViewState(
            email = "test",
            password = PasswordViewState(password = "test")
        ),
        onEmailTextChange = {},
        onPasswordTextChange = {},
        onPasswordVisibilityClick = {},
        onSignInClicked = {}
    )
}
```

## AddFoodEstablishmentsScreenViewModel.kt

```
package com.project.presentation.ui.screens.add_food_establishments


@HiltViewModel
class AddFoodEstablishmentsScreenViewModel @Inject constructor(
    private val foodEstablishmentRepository: FoodEstablishmentRepository,
    private val getCurrentUserUseCase: GetCurrentUserUseCase
) : ViewModel() {

    private val _uiState: MutableStateFlow<AddFoodEstablishmentsUIState> =
        MutableStateFlow(AddFoodEstablishmentsUIState())
    val uiState: StateFlow<AddFoodEstablishmentsUIState> =
_uiState.asStateFlow()
```

```
fun onMainInfoNameChanged(name: String) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                name = name
            )
        )
    }
}

fun onMainInfoTypeChanged(foodEstablishmentType: FoodEstablishmentType) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                foodEstablishmentType = foodEstablishmentType
            )
        )
    }
}

fun onMainInfoAddressChanged(address: String) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                address = address
            )
        )
    }
}

fun onMainInfoCityChanged(city: String) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                city = city
            )
        )
    }
}

fun onMainInfoDescriptionChanged(description: String) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                description = description
            )
        )
    }
}

fun onMainInfoTimeFromSelected(time: Long) {
    _uiState.update {
        it.copy(
            mainInfoViewState = it.mainInfoViewState.copy(
                selectedTimeFrom = time,
                selectedTimeTo = null
            )
        )
    }
}

fun onMainInfoTimeToSelected(time: Long) {
    _uiState.update {
        it.copy(
```

```
                mainInfoViewState = it.mainInfoViewState.copy(
                    selectedTimeTo = time
                )
            )
        }
    }

    fun onMainInfoPhoneForReservationChanged(phone: String) {
        _uiState.update {
            it.copy(
                mainInfoViewState = it.mainInfoViewState.copy(
                    phoneForReservation = phone
                )
            )
        }
    }

    fun addTagToSelected(tag: Tag) {
        val newTag = Tag(
            title = tag.title,
            isSelected = true
        )
        val newSelectedList =
_uiState.value.addTagsViewState.selectedTagsList.toMutableList()
        newSelectedList.add(newTag)
        val newRecommendedList =
_uiState.value.addTagsViewState.recommendedTagsList.toMutableList()
        newRecommendedList.remove(tag)
        _uiState.update {
            it.copy(
                addTagsViewState = AddTagsViewState(
                    selectedTagsList = newSelectedList,
                    recommendedTagsList = newRecommendedList
                )
            )
        }
    }

    fun removeTagFromSelected(tag: Tag) {
        val newTag = Tag(
            title = tag.title,
            isSelected = false
        )
        val newSelectedList =
_uiState.value.addTagsViewState.selectedTagsList.toMutableList()
        newSelectedList.remove(tag)
        val newRecommendedList =
_uiState.value.addTagsViewState.recommendedTagsList.toMutableList()
        val isTagWasInitiallyRecommended = Tags.values().map { it.title
}.contains(newTag.title)

        if (isTagWasInitiallyRecommended) {
            newRecommendedList.add(newTag)
        }
        _uiState.update {
            it.copy(
                addTagsViewState = AddTagsViewState(
                    selectedTagsList = newSelectedList,
                    recommendedTagsList = newRecommendedList
                )
            )
        }
    }
```

```
    fun setValueForAddNewTagClicked(value: Boolean) {
        _uiState.update {
            it.copy(
                addTagsViewState = it.addTagsViewState.copy(
                    addNewTagClicked = value
                )
            )
        }
    }

    fun onTablesCounterChanged(value: Int) {
        _uiState.update {
            it.copy(
                setTablesCounterViewState = it.setTablesCounterViewState.copy(
                    tablesCount = value
                )
            )
        }
    }

    fun onContinueClicked() {
        /**
         * The stepNumber of currentStep equal to index of next step
         */
        val nextStep =

AddFoodEstablishmentStep.values()[_uiState.value.currentStep.stepNumber]
        _uiState.update {
            it.copy(
                currentStep = nextStep
            )
        }
    }

    fun decreaseStepNumber() {
        val nextStepNumber = _uiState.value.currentStep.stepNumber - 1
        AddFoodEstablishmentStep.values().firstOrNull {
            it.stepNumber == nextStepNumber
        }?.let {
            _uiState.update {
                it.copy(
                    currentStep = it.currentStep
                )
            }
        }
    }

    fun changePhoto(index: Int, uri: Uri) {
        val newList = _uiState.value.addPhotoViewState.photoList.toMutableList()
        newList[index] = Photo(index, uri)
        _uiState.update {
            it.copy(
                addPhotoViewState = AddPhotoViewState(photoList = newList)
            )
        }
    }

    fun onRegisterClicked() {
        viewModelScope.launch {
            _uiState.update {
                it.copy(
                    isLoading = true
                )
            }
```

```
            val idOfNewFoodEstablishment = UUID.randomUUID().toString()

            val selectedTimeFrom =
_uiState.value.mainInfoViewState.selectedTimeFrom!!
            val selectedTimeTo =
_uiState.value.mainInfoViewState.selectedTimeTo!!
            val tablesCount =
_uiState.value.setTablesCounterViewState.tablesCount
            val generatedTables = generateTables(tablesCount, selectedTimeFrom,
selectedTimeTo)
            foodEstablishmentRepository.registerFoodEstablishment(
                foodEstablishment = FoodEstablishment(
                    id = idOfNewFoodEstablishment,
                    name = _uiState.value.mainInfoViewState.name ?: "",
                    foodEstablishmentType =
_uiState.value.mainInfoViewState.foodEstablishmentType!!,
                    address = _uiState.value.mainInfoViewState.address ?: "",
                    description = _uiState.value.mainInfoViewState.description
?: "",
                    photoList = _uiState.value.addPhotoViewState.photoList,
                    ownerName =
getCurrentUserUseCase.invoke().getOrNull()?.nameSurname ?: "",
                    city = _uiState.value.mainInfoViewState.city ?: "",
                    selectedTimeTo =
_uiState.value.mainInfoViewState.selectedTimeTo,
                    selectedTimeFrom =
_uiState.value.mainInfoViewState.selectedTimeFrom,
                    phoneForBooking =
_uiState.value.mainInfoViewState.phoneForReservation ?: "",
                    tags = _uiState.value.addTagsViewState.selectedTagsList.map
{ it.title },
                    tablesForBooking = generatedTables
                )
            ).fold(
                onSuccess = {
                    _uiState.update {
                        it.copy(
                            isLoading = false,
                            navigateToProfile = true
                        )
                    }
                },
                onFailure = {
                    _uiState.update {
                        it.copy(
                            isLoading = false,
                            isError = true
                        )
                    }
                }
            )
        }
    }

    private fun generateTables(count: Int, timeFrom: Long, timeTo: Long):
List<Table> {
        val tables = mutableListOf<Table>()
        for (i in 0 until count) {
            val slotsList = mutableListOf<TimeSlot>()
            val calendarFrom = Calendar.getInstance().apply {
                timeInMillis = timeFrom
            }
            val calendarTo = Calendar.getInstance().apply {
                timeInMillis = timeTo
```

```
            }
            for (j in 0 until DAYS_AHEAD) {
                val slots = generateTimeSlots(calendarFrom.timeInMillis,
calendarTo.timeInMillis)
                slotsList.addAll(slots)
                calendarFrom.add(Calendar.DAY_OF_MONTH, 1)
                calendarTo.add(Calendar.DAY_OF_MONTH, 1)
            }
            tables.add(Table(timeSlots = slotsList))
        }

        return tables
    }

    private fun generateTimeSlots(timeFrom: Long, timeTo: Long): List<TimeSlot>
{
        val timeSlots = mutableListOf<TimeSlot>()
        var currentTime = timeFrom
        var nextTime = currentTime + INTERVAL
        while (currentTime < timeTo) {
            if (nextTime <= timeTo) {
                timeSlots.add(TimeSlot(currentTime, nextTime))
            }
            currentTime = nextTime
            nextTime = currentTime + INTERVAL
        }
        return timeSlots
    }

    companion object {
        private const val INTERVAL = 30 * 60 * 1000 // 1 hour in milliseconds
        private const val DAYS_AHEAD = 30
    }
}

data class AddFoodEstablishmentsUIState(
    val isLoading: Boolean = false,
    val currentStep: AddFoodEstablishmentStep =
AddFoodEstablishmentStep.AddFoodEstablishmentMainInfo,
    val mainInfoViewState: MainInfoViewState = MainInfoViewState(),
    val addTagsViewState: AddTagsViewState = AddTagsViewState(),
    val setTablesCounterViewState: SetTablesCounterViewState =
SetTablesCounterViewState(),
    val addPhotoViewState: AddPhotoViewState = AddPhotoViewState(),
    val navigateToProfile: Boolean = false,
    val isError: Boolean = false
) {
    private fun getMaxSteps() = 4

    fun getProgress() =
        if (currentStep.stepNumber == getMaxSteps()) 1f else
currentStep.stepNumber * 0.3f

    fun getTitle(): String {
        return "${currentStep.stepNumber}/${getMaxSteps()} ${currentStep.title}"
    }
}
```

## HomeViewModel.kt

```
package com.project.presentation.ui.screens.home
```

```
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.project.domain.model.FoodEstablishment
import com.project.domain.repository.FoodEstablishmentRepository
import com.project.domain.repository.SelectedDateForBookingLocalRepository
import com.project.presentation.ui.view.HomeSearchViewState
import com.project.presentation.ui.view.register_food_establishment.Tag
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import timber.log.Timber
import javax.inject.Inject

@HiltViewModel
class HomeViewModel @Inject constructor(
    private val foodEstablishmentRepository: FoodEstablishmentRepository
) : ViewModel() {

    private val _uiState: MutableStateFlow<HomeUIState> =
        MutableStateFlow(HomeUIState())
    val uiState: StateFlow<HomeUIState> = _uiState.asStateFlow()

    init {
        viewModelScope.launch {
            _uiState.update {
                it.copy(
                    isLoading = true
                )
            }
            foodEstablishmentRepository.fetchFoodEstablishments()
                .fold(
                    onSuccess = {
                        val tagsTitleList = mutableListOf<String>()
                        it.forEach {
                            it.tags.forEach {
                                if (!tagsTitleList.contains(it)) {
                                    tagsTitleList.add(it)
                                }
                            }

                        }
                        val tags = tagsTitleList.map {
                            Tag(
                                title = it
                            )
                        }
                        _uiState.update {
                            it.copy(
                                homeSearchViewState =
it.homeSearchViewState.copy(
                                    tags = tags
                                )
                            )
                        }
                    },
                    onFailure = {
                        Timber.e(it)
                    }
                )
            _uiState.update {
                it.copy(
```

```
                    isLoading = false
                )
            }
        }
    }

    fun onCityChanged(city: String) {
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
                    city = city
                )
            )
        }
    }

    fun handleTagSelection(tag: Tag) {
        val index = _uiState.value.homeSearchViewState.tags.indexOf(tag)
        val newTag = Tag(
            title = tag.title,
            isSelected = !tag.isSelected
        )
        val newSelectedList =
_uiState.value.homeSearchViewState.tags.toMutableList()
        newSelectedList.removeAt(index)
        newSelectedList.add(index, newTag)
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
                    tags = newSelectedList
                )
            )
        }
    }

    fun onSearchClicked() {
        val date = _uiState.value.homeSearchViewState.selectedDate!!
        SelectedDateForBookingLocalRepository.saveData(
            date = date,
            selectedTimeFrom =
_uiState.value.homeSearchViewState.selectedTimeFrom!!,
            selectedTimeTo =
_uiState.value.homeSearchViewState.selectedTimeTo!!,
            peopleCount = _uiState.value.homeSearchViewState.peopleCount
        )
        _uiState.update {
            it.copy(
                continueClicked = true
            )
        }
    }

    fun resetContinueClickedStatus() {
        _uiState.update {
            it.copy(
                continueClicked = false
            )
        }
    }

    fun onDateChanged(value: Long) {
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
```

```
                    selectedDate = value
                )
            )
        }
    }

    fun onPeopleCounterChanged(value: Int) {
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
                    peopleCount = value
                )
            )
        }
    }


    fun onStartTimeChanged(time: Long) {
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
                    selectedTimeFrom = time,
                    selectedTimeTo = null
                )
            )
        }
    }

    fun onEndTimeChanged(time: Long) {
        _uiState.update {
            it.copy(
                homeSearchViewState = it.homeSearchViewState.copy(
                    selectedTimeTo = time
                )
            )
        }
    }
}

data class HomeUIState(
    val isLoading: Boolean = false,
    val list: List<FoodEstablishment> = emptyList(),
    val homeSearchViewState: HomeSearchViewState = HomeSearchViewState(),
    val continueClicked: Boolean = false
)
```

## StatisticsScreen.kt

```
package com.project.presentation.ui.screens.statistic

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
```

```
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.text.SpanStyle
import androidx.compose.ui.text.buildAnnotatedString
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.withStyle
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.lifecycle.compose.collectAsStateWithLifecycle
import com.github.tehras.charts.bar.BarChart
import com.github.tehras.charts.bar.BarChartData
import com.github.tehras.charts.bar.renderer.bar.SimpleBarDrawer
import com.github.tehras.charts.bar.renderer.label.SimpleValueDrawer
import com.github.tehras.charts.bar.renderer.xaxis.SimpleXAxisDrawer
import com.github.tehras.charts.bar.renderer.yaxis.SimpleYAxisDrawer
import com.github.tehras.charts.piechart.PieChart
import com.github.tehras.charts.piechart.PieChartData
import com.github.tehras.charts.piechart.animation.simpleChartAnimation
import com.github.tehras.charts.piechart.renderer.SimpleSliceDrawer
import com.project.domain.repository.SurveyData
import com.project.presentation.R
import com.project.presentation.ui.view.common.LoadingView
import kotlin.math.roundToInt

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun StatisticsScreen(
    modifier: Modifier = Modifier,
    viewModel: StatisticsScreenViewModel = hiltViewModel(),
    navigateBack: () -> Unit,
) {

    val uiState by viewModel.uiState.collectAsStateWithLifecycle()

    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = {
                    Text(
                        text = "Статистика закладу",
                        style = MaterialTheme.typography.titleMedium.copy(color
= Color.White)
                    )
                },
                navigationIcon = {
                    IconButton(
                        onClick = {
                            navigateBack()
                        }) {
```

```
                        Icon(
                            imageVector = Icons.Filled.ArrowBack,
                            tint = Color.White,
                            contentDescription = null
                        )
                    }
                },
                colors = TopAppBarDefaults.smallTopAppBarColors(containerColor =
Color.Black)
            )
        }
    ) { contentPadding ->
        Box(
            modifier = modifier
                .padding(contentPadding)
                .fillMaxSize(),
        ) {
            if (uiState.isLoading) {
                LoadingView()
            } else {
                Box(modifier = Modifier.fillMaxSize()) {
                    Column(
                        modifier = Modifier
                            .verticalScroll(rememberScrollState())
                            .padding(horizontal = 20.dp)
                    ) {

                        Spacer(modifier = Modifier.height(20.dp))
                        Text(
                            buildAnnotatedString {
                                append("На питання ")
                                withStyle(
                                    style = SpanStyle(
                                        color = Color.Black,
                                        fontWeight = FontWeight.Bold
                                    )
                                ) {
                                    append(SurveyData.getData()[0].question)
                                }
                                append(" користувачі відповіли:")
                            }
                        )
                        val totalFirst = uiState.firstQuestionPair?.first?.plus(
                            uiState.firstQuestionPair?.second ?: 0
                        ) ?: 0
                        val percentageYesFirst =

(uiState.firstQuestionPair?.first?.toDouble()?.div(totalFirst))?.times(
                                100
                            )
                        val percentageNoFirst =
(uiState.firstQuestionPair?.second?.toDouble()
                            ?.div(totalFirst))?.times(100)

                        Spacer(modifier = Modifier.height(10.dp))
                        PieChart(
                            pieChartData = PieChartData(
                                listOf(
                                    PieChartData.Slice(
                                        percentageYesFirst?.toFloat() ?: 0f,
                                        colorResource(id = R.color.green)
                                    ),
                                    PieChartData.Slice(
                                        percentageNoFirst?.toFloat() ?: 0f,
```

```
                                    colorResource(id = R.color.red)
                                )
                            )
                        ),
                        modifier = Modifier.size(200.dp),
                        animation = simpleChartAnimation(),
                        sliceDrawer = SimpleSliceDrawer()
                    )
                    Spacer(modifier = Modifier.height(10.dp))
                    AnswersDescription(percentageYesFirst,
percentageNoFirst)
                    Spacer(modifier = Modifier.height(20.dp))
                    Text(
                        buildAnnotatedString {
                            append("На питання ")
                            withStyle(
                                style = SpanStyle(
                                    color = Color.Black,
                                    fontWeight = FontWeight.Bold
                                )
                            ) {
                                append(SurveyData.getData()[1].question)
                            }
                            append(" користувачі відповіли:")
                        }
                    )
                    val totalSecond =
uiState.secondQuestionPair?.first?.plus(
                        uiState.secondQuestionPair?.second ?: 0
                    ) ?: 0
                    val percentageYesSecond =
                        (uiState.secondQuestionPair?.first?.toDouble()
                            ?.div(totalSecond))?.times(
                                100
                            )
                    val percentageNoSecond =
(uiState.secondQuestionPair?.second?.toDouble()
                        ?.div(totalSecond))?.times(100)

                    Spacer(modifier = Modifier.height(10.dp))
                    PieChart(
                        pieChartData = PieChartData(
                            listOf(
                                PieChartData.Slice(
                                    percentageYesSecond?.toFloat() ?: 0f,
                                    colorResource(id = R.color.green)
                                ),
                                PieChartData.Slice(
                                    percentageNoSecond?.toFloat() ?: 0f,
                                    colorResource(id = R.color.red)
                                )
                            )
                        ),
                        modifier = Modifier.size(200.dp),
                        animation = simpleChartAnimation(),
                        sliceDrawer = SimpleSliceDrawer()
                    )

                    Spacer(modifier = Modifier.height(10.dp))
                    AnswersDescription(percentageYesSecond,
percentageNoSecond)
                    Spacer(modifier = Modifier.height(20.dp))
                    Text(
                        buildAnnotatedString {
```

```
                                append("На питання ")
                                withStyle(
                                    style = SpanStyle(
                                        color = Color.Black,
                                        fontWeight = FontWeight.Bold
                                    )
                                ) {
                                    append(SurveyData.getData()[2].question)
                                }
                                append(" користувачі відповіли:")
                            }
                        )
                        Spacer(modifier = Modifier.height(10.dp))
                        BarChart(
                            barChartData = BarChartData(
                                bars = listOf(
                                    BarChartData.Bar(
                                        label = "1",
                                        value = uiState.thirdList[0],
                                        color = colorResource(id =
R.color.violet)
                                    ),
                                    BarChartData.Bar(
                                        label = "2",
                                        value = uiState.thirdList[1],
                                        color = colorResource(id = R.color.blue)
                                    ),
                                    BarChartData.Bar(
                                        label = "3",
                                        value = uiState.thirdList[2],
                                        color = colorResource(id =
R.color.malin)
                                    ),
                                    BarChartData.Bar(
                                        label = "4",
                                        value = uiState.thirdList[3],
                                        color = colorResource(id =
R.color.light_green)
                                    ),
                                    BarChartData.Bar(
                                        label = "5",
                                        value = uiState.thirdList[4],
                                        color = colorResource(id =
R.color.coral)
                                    )
                                )
                            ),
                            modifier = Modifier
                                .fillMaxSize()
                                .height(250.dp),
                            animation = simpleChartAnimation(),
                            barDrawer = SimpleBarDrawer(),
                            xAxisDrawer = SimpleXAxisDrawer(),
                            yAxisDrawer = SimpleYAxisDrawer(
                                labelRatio = 100,
                                labelValueFormatter = {
                                    it.toInt().toString()
                                }
                            ),
                            labelDrawer = SimpleValueDrawer(
                                drawLocation =
SimpleValueDrawer.DrawLocation.XAxis,
                            )
                        )
```

```
                        Spacer(modifier = Modifier.height(20.dp))
                        Column {
                            TextDescriptionForBarChart(
                                number = "1",
                                value = SurveyData.getData()[2].answers[0]
                            )
                            Spacer(modifier = Modifier.height(6.dp))
                            TextDescriptionForBarChart(
                                number = "2",
                                value = SurveyData.getData()[2].answers[1]
                            )
                            Spacer(modifier = Modifier.height(6.dp))
                            TextDescriptionForBarChart(
                                number = "3",
                                value = SurveyData.getData()[2].answers[2]
                            )
                            Spacer(modifier = Modifier.height(6.dp))
                            TextDescriptionForBarChart(
                                number = "4",
                                value = SurveyData.getData()[2].answers[3]
                            )
                            Spacer(modifier = Modifier.height(6.dp))
                            TextDescriptionForBarChart(
                                number = "5",
                                value = SurveyData.getData()[2].answers[4]
                            )
                        }
                        Spacer(modifier = Modifier.height(30.dp))
                    }
                }
            }
        }
    }
}

@Composable
private fun TextDescriptionForBarChart(
    number: String,
    value: String
) {
    Text(
        buildAnnotatedString {
            withStyle(
                style = SpanStyle(
                    color = Color.Black,
                    fontWeight = FontWeight.Bold,
                    fontSize = 12.sp
                )
            ) {
                append(number)
            }
            append(" - $value")
        }
    )
}

@Composable
private fun AnswersDescription(percentageYes: Double?, percentageNo: Double?) {
    Row(
        verticalAlignment = Alignment.CenterVertically,
    ) {
        Box(
            modifier = Modifier
                .size(20.dp)
```

```
                .clip(RoundedCornerShape(6.dp))
                .background(colorResource(id = R.color.green))
        )
        Spacer(modifier = Modifier.size(6.dp))
        Text(
            text = "Так (${percentageYes?.roundToInt()}%)",
            style = MaterialTheme.typography.bodySmall.copy(color = Color.Black)
        )
        Spacer(modifier = Modifier.size(10.dp))
        Box(
            modifier = Modifier
                .size(20.dp)
                .clip(RoundedCornerShape(6.dp))
                .background(colorResource(id = R.color.red))
        )
        Spacer(modifier = Modifier.size(6.dp))
        Text(
            text = "Ні (${percentageNo?.roundToInt()}%)",
            style = MaterialTheme.typography.bodySmall.copy(color = Color.Black)
        )
    }
}
```

StatisticsScreenViewModel.kt

```
package com.project.presentation.ui.screens.statistic

import androidx.lifecycle.SavedStateHandle
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.project.domain.repository.FoodEstablishmentRepository
import com.project.domain.repository.SurveyData
import com.project.presentation.ui.navigation.ArgsName
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import kotlinx.coroutines.flow.update
import kotlinx.coroutines.launch
import timber.log.Timber
import javax.inject.Inject

@HiltViewModel
class StatisticsScreenViewModel @Inject constructor(
    foodEstablishmentRepository: FoodEstablishmentRepository,
    savedStateHandle: SavedStateHandle
) : ViewModel() {

    private val id: String = checkNotNull(savedStateHandle[ArgsName.ID_ARG])

    private val _uiState: MutableStateFlow<StatisticsScreenUiModel> =
        MutableStateFlow(StatisticsScreenUiModel())
    val uiState: StateFlow<StatisticsScreenUiModel> = _uiState.asStateFlow()

    init {
        viewModelScope.launch {
            _uiState.update {
                it.copy(isLoading = true)
            }
            foodEstablishmentRepository.getFoodEstablishmentById(id).fold(
                onSuccess = {
                    var firstQuestionYes = 0
                    var firstQuestionNo = 0
                    var secondQuestionYes = 0
```

```
                    var secondQuestionNo = 0
                    var cuisineCounter = 0
                    var serviceCounter = 0
                    var atmosphereCounter = 0
                    var locationCounter = 0
                    var nothing = 0
                    it.statisticModelList.forEach { statistic ->
                        statistic.map?.forEach { (taskIndex, answer) ->
                            val taskIndexInt = taskIndex.toInt()
                            if (taskIndexInt == 0) {
                                if (answer ==
SurveyData.getData()[0].answers[0]) {
                                    firstQuestionYes++
                                } else {
                                    firstQuestionNo++
                                }
                            } else if (taskIndexInt == 1) {
                                if (answer ==
SurveyData.getData()[1].answers[0]) {
                                    secondQuestionYes++
                                } else {
                                    secondQuestionNo++
                                }
                            } else if (taskIndexInt == 2) {
                                when (answer) {
                                    SurveyData.getData()[2].answers[0] ->
cuisineCounter++
                                    SurveyData.getData()[2].answers[1] ->
serviceCounter++
                                    SurveyData.getData()[2].answers[2] ->
atmosphereCounter++
                                    SurveyData.getData()[2].answers[3] ->
locationCounter++
                                    else -> nothing++
                                }
                            }
                        }
                    }
                    _uiState.update {
                        it.copy(
                            firstQuestionPair = Pair(firstQuestionYes,
firstQuestionNo),
                            secondQuestionPair = Pair(secondQuestionYes,
secondQuestionNo),
                            thirdList = listOf(
                                cuisineCounter.toFloat(),
                                serviceCounter.toFloat(),
                                atmosphereCounter.toFloat(),
                                locationCounter.toFloat(),
                                nothing.toFloat()
                            )
                        )
                    }
                },
                onFailure = {
                    Timber.e(it)
                }
            )
            _uiState.update {
                it.copy(
                    isLoading = false,
                )
            }
        }
```

```
    }
}

data class StatisticsScreenUiModel(
    val isLoading: Boolean = false,
    val firstQuestionPair: Pair<Int, Int>? = null,
    val secondQuestionPair: Pair<Int, Int>? = null,
    val thirdList: List<Float> = emptyList()
)
```

MyReservationsScreen.kt

```
package com.project.presentation.ui.screens.myreservations

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.lifecycle.compose.collectAsStateWithLifecycle
import com.airbnb.lottie.compose.LottieAnimation
import com.airbnb.lottie.compose.LottieCompositionSpec
import com.airbnb.lottie.compose.LottieConstants
import com.airbnb.lottie.compose.animateLottieCompositionAsState
import com.airbnb.lottie.compose.rememberLottieComposition
import com.project.presentation.R
import
com.project.presentation.ui.screens.myreservations.view.MyReservationItemView
import com.project.presentation.ui.view.common.EmptyListView
import com.project.presentation.ui.view.common.LoadingView

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyReservationsScreen(
    modifier: Modifier = Modifier,
    viewModel: MyReservationsScreenViewModel = hiltViewModel(),
    navigateToPdp: (String) -> Unit,
) {
    val uiState by viewModel.uiState.collectAsStateWithLifecycle()

    Scaffold(
        topBar = {
```

```
            TopAppBar(
                title = {
                    Text(
                        text = stringResource(R.string.reservations),
                        style = MaterialTheme.typography.titleMedium.copy(color
= Color.White)
                    )
                },
                colors = TopAppBarDefaults.smallTopAppBarColors(containerColor =
Color.Black)
            )
        }
    ) { contentPadding ->
        Box(
            modifier = modifier
                .padding(contentPadding)
                .fillMaxSize(),
        ) {
            if (uiState.isLoading) {
                LoadingView()
            } else {
                Box(
                    modifier = Modifier
                        .fillMaxSize()
                ) {
                    if (uiState.reservations.isEmpty()) {
                        EmptyListView(text = "Жодного бронювання не було
знайдено")
                    } else {
                        LazyColumn(
                            modifier = Modifier.padding(20.dp)
                        ) {
                            items(uiState.reservations) { item ->
                                MyReservationItemView(
                                    viewState = item,
                                    removeReservation = {
                                        viewModel.removeReservation(
                                            foodEstablishmentId =
item.foodEstablishmentId,

                                            timeSlot = item.timeSlot
                                        )
                                    },
                                    navigateToPdp = {
                                        navigateToPdp(item.foodEstablishmentId)
                                    }
                                )
                                Spacer(modifier = Modifier.height(20.dp))
                            }
                        }
                        Spacer(modifier = Modifier.height(80.dp))
                    }
                }
            }
        }
    }
}
```

SetReservationScreen.kt

```
package com.project.presentation.ui.screens.setreservation

import androidx.annotation.StringRes
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
```

```
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material.icons.filled.Info
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.CenterAlignedTopAppBar
import androidx.compose.material3.Divider
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.lifecycle.compose.collectAsStateWithLifecycle
import com.project.presentation.R
import com.project.presentation.ui.view.common.LoadingView

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SetReservationScreen(
    modifier: Modifier = Modifier,
    viewModel: SetReservationScreenViewModel = hiltViewModel(),
    navigateBack: () -> Unit
) {

    val uiState by viewModel.uiState.collectAsStateWithLifecycle()
    if (uiState.navigateBack) {
        navigateBack()
    }
    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = {
                    Text(
                        text = stringResource(R.string.resto_booking),
                        style = MaterialTheme.typography.titleMedium.copy(color
= Color.White)
                    )
                },
```

```
                navigationIcon = {
                    IconButton(
                        onClick = {
                            navigateBack()
                        }) {
                        Icon(
                            imageVector = Icons.Filled.ArrowBack,
                            tint = Color.White,
                            contentDescription = null
                        )
                    }
                },
                colors = TopAppBarDefaults.smallTopAppBarColors(containerColor =
Color.Black)
            )
        }
    ) { contentPadding ->
        Box(
            modifier = modifier
                .padding(contentPadding)
                .fillMaxSize(),
        ) {
            if (uiState.isLoading) {
                LoadingView()
            } else {
                Box(
                    modifier = Modifier
                        .fillMaxSize()
                ) {
                    Column(
                        modifier = Modifier
                            .padding(20.dp)
                            .verticalScroll(rememberScrollState())
                            .align(Alignment.TopCenter)
                    ) {

                        InfoMessage(messageRes = R.string.comment_message)
                        Spacer(modifier = Modifier.height(20.dp))
                        OutlinedTextField(
                            value = uiState.comment ?: "",
                            modifier = Modifier
                                .fillMaxWidth(),
                            onValueChange = {
                                viewModel.onCommentChanged(it)
                            },
                            placeholder = { Text(text = stringResource(id =
R.string.comment)) },
                            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Text),
                            colors = TextFieldDefaults.outlinedTextFieldColors(
                                cursorColor = colorResource(id = R.color.gray),
                                textColor = Color.Black,
                                placeholderColor = colorResource(id =
R.color.gray),
                                focusedBorderColor = Color.Black,
                                unfocusedBorderColor = Color.Black
                            ),
                            shape = RoundedCornerShape(8.dp),
                        )
                    }
                    Column(
                        modifier = Modifier
                            .fillMaxWidth()
                            .align(Alignment.BottomCenter)
```

```
                    ) {
                        Divider(
                            color = colorResource(id = R.color.main_yellow),
                            thickness = 1.dp
                        )
                        Box(
                            modifier = Modifier
                                .fillMaxWidth()
                                .background(Color.White)
                                .padding(horizontal = 20.dp, vertical = 8.dp),
                        ) {
                            Button(
                                modifier = Modifier
                                    .align(Alignment.Center),
                                onClick = {
                                    viewModel.finishReservation()
                                },
                                shape = RoundedCornerShape(8.dp),
                                colors = ButtonDefaults.buttonColors(
                                    containerColor = colorResource(
                                        id = R.color.main_yellow
                                    )
                                )
                            ) {
                                Text(
                                    modifier = Modifier
                                        .fillMaxWidth(),
                                    textAlign = TextAlign.Center,
                                    text = "Завершити бронювання",
                                    style = MaterialTheme.typography.titleMedium
                                )
                            }
                        }
                    }
                }
            }
        }
    }
}

@Composable
private fun InfoMessage(
    modifier: Modifier = Modifier,
    @StringRes messageRes: Int
) {
    Row(
        modifier = modifier,
        verticalAlignment = Alignment.CenterVertically
    ) {
        Icon(
            imageVector = Icons.Default.Info,
            tint = colorResource(id = R.color.main_yellow),
            contentDescription = null
        )
        Spacer(modifier = Modifier.width(14.dp))
        Text(
            text = stringResource(messageRes),
            style = MaterialTheme.typography.bodyLarge
        )
    }
}
```

# Додаток Г – Ілюстративна частина

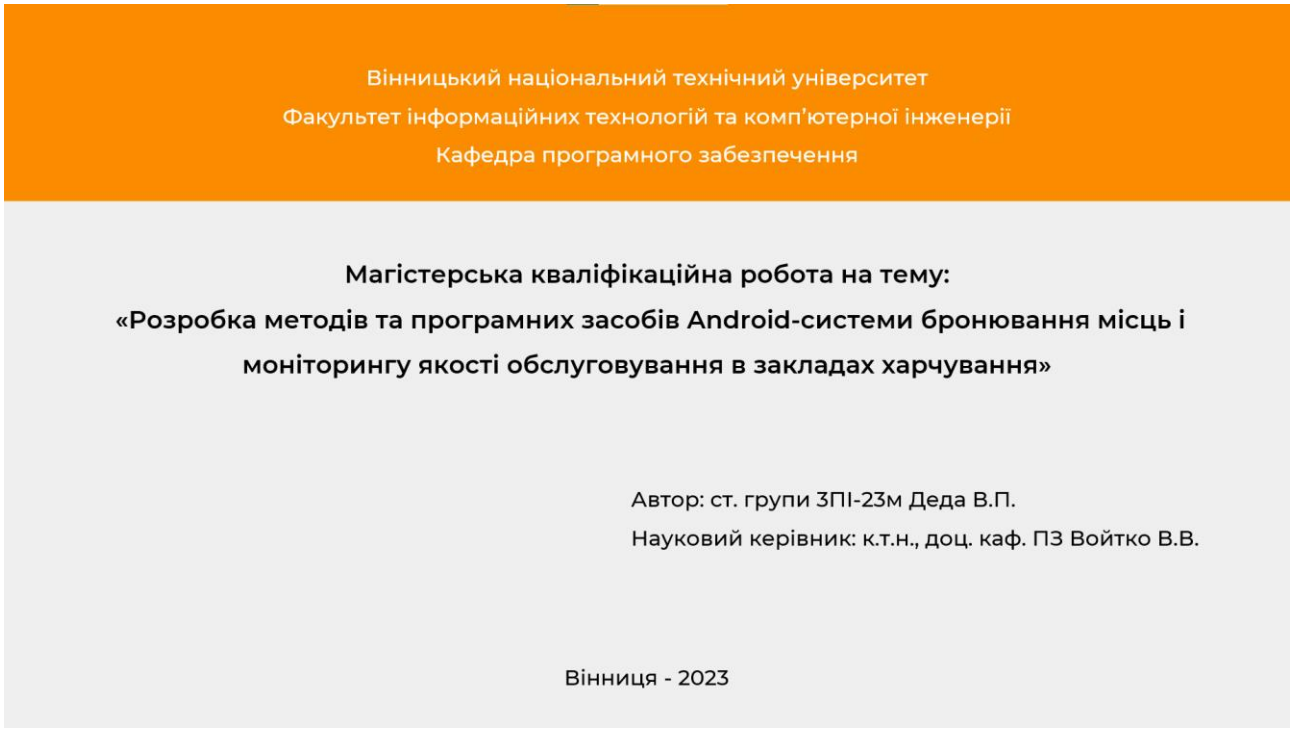



Рисунок Г.1 – Назва роботи


Рисунок Г.2 – Мета, об’єкт та предмет роботи

## Завдання

- визначити засоби реалізації мобільної системи бронювання місць і моніторингу якості обслуговування в закладах харчування;
- розробити метод додавання нового закладу в систему;
- розробити метод пошуку закладів харчування відповідно до обраного міста і обраних тегів;
- розробити метод додавання відгуків і рейтингу про заклади харчування;
- розробити метод бронювання місць у закладах харчування;
- розробити усі програмні модулі компонентів системи мобільного застосунку;
- провести тестування розробленої системи.

Рисунок Г.3 – Завдання роботи

## Наукова новизна

- Подальшогорозвитку отримав метод пошуку закладів харчування для бронювання місць, який, на відміну від існуючих, має персоналізований підхід до пошуку та застосовує адаптивні алгоритми з індивідуалізацією пошукових процесів, що дозволяє покращити інтерактивні комунікації з клієнтами.
- Подальшого розвитку отримав метод моніторингу якості обслуговування в закладах харчування, який, на відміну від існуючих, реалізує в середовищі мобільної системи спеціалізовані засоби підтримки інтерактивної взаємодії працівників з клієнтами, що впливає на визначення статусу закладу харчування в системі рейтингового оцінювання в процесі формування списку пріоритетних варіантів і сприяє удосконаленнюбізнес-стратегії закладів харчування.

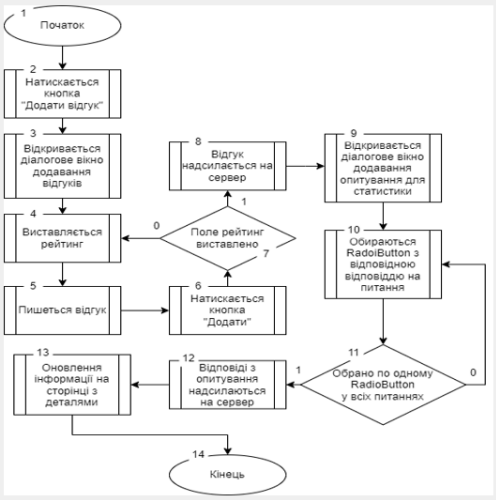
Рисунок Г.4 – Наукова новизна

## Практична цінність отриманих результатів

Розроблена Android-система може використовуватися закладами харчування для пришвидшення процесу бронювання місць, а також для отримання зворотного зв'язку про якість обслуговування.

Рисунок Г.5 – Практична цінність отриманих

## Метод моніторингу якості обслуговування




1.Натискається кнопка «Додати відгук» ClickListener якої, в свою чергу відкриває діалогове вікно додавання відгуків.
2.На RatingBar виставляється рейтинг від 1 до 5.
3.У полі TextField з inputType Text пишеться відгук.
4.Статус enabled у кнопки стає true.
5.Натискається кнопка «Додати».
6.Відгук надсилається на сервер методом POST.
7.Відкривається діалогове вікно опитування для статистики про заклад.
8.Обираються RadioButton для вибору відповіді на питання.
9.MutableStateFlow оновлюється і параметр кнопки «Додати» стає true.
10.Опитування надсилається на Firebase і додається до статистики закладу.
11.Оновлюється інформація про заклад і доданий відгук відображається у загальному списку усіх відгуків

Рисунок Г.6 – Метод моніторингу якості обслуговування


Рисунок Г.7 – Метод додавання нового закладу харчування в систему (перша частина методу)


Рисунок Г.8 – Метод додавання нового закладу харчування в систему (друга частина методу)

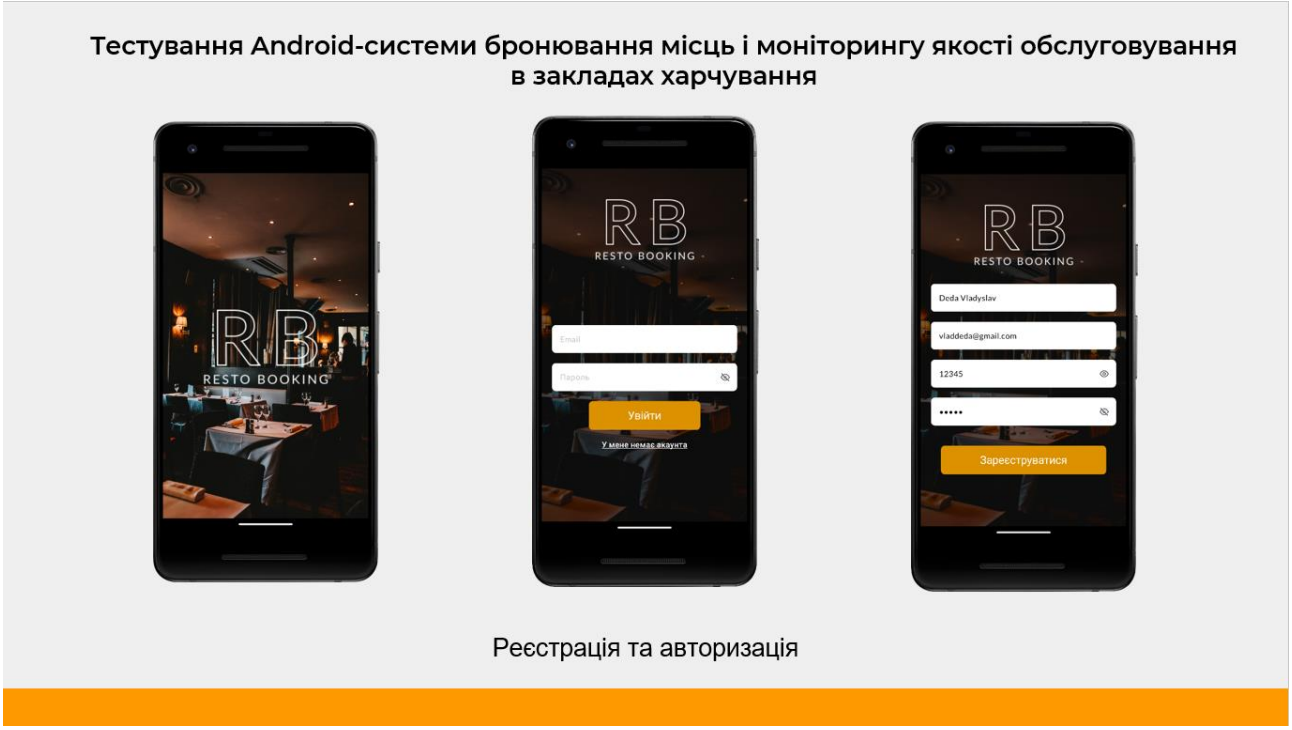



Рисунок Г.9 – Реєстрація та авторизація

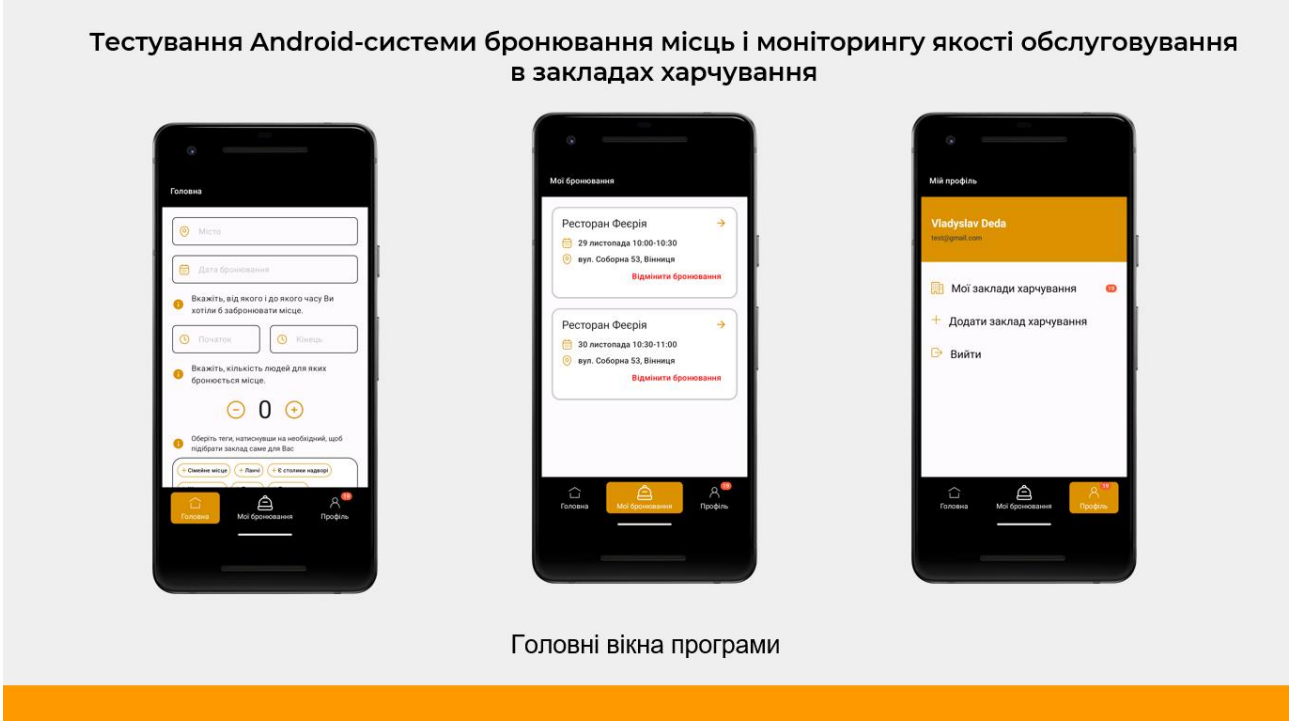



Рисунок Г.10 – Головні вікна програми

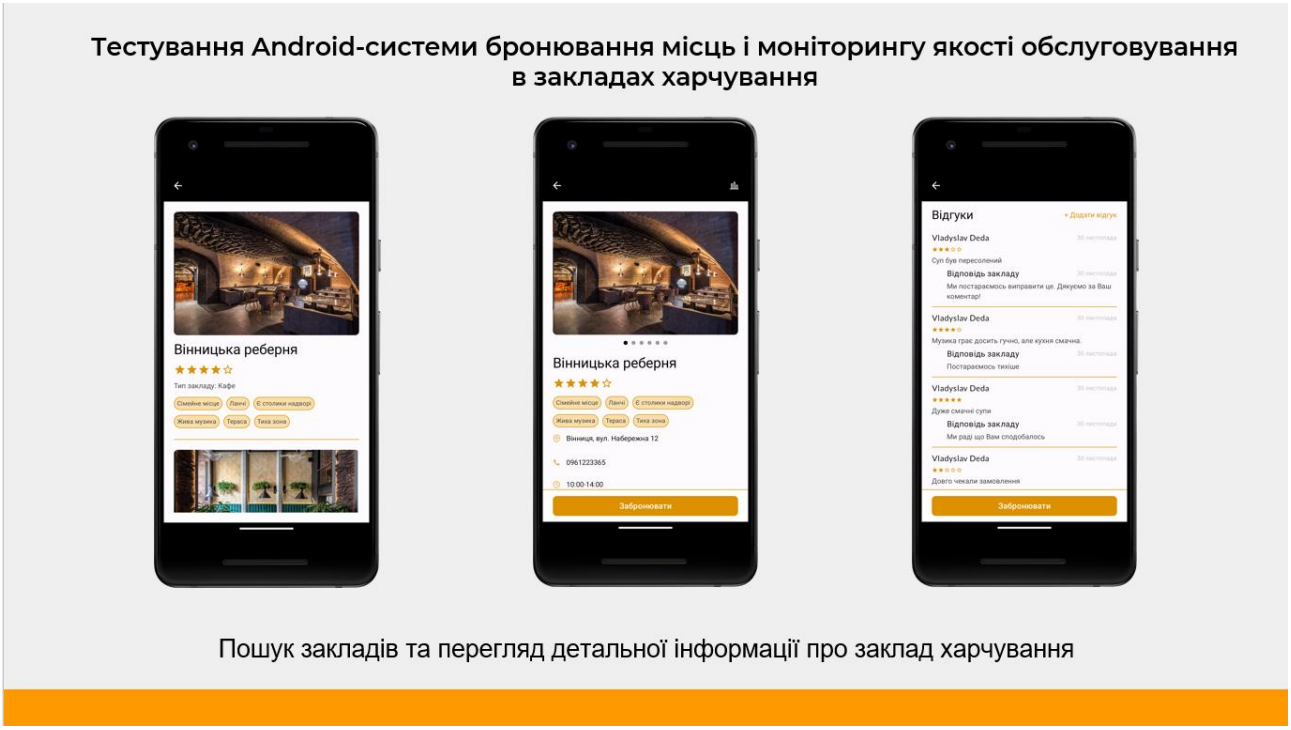



Рисунок Г.11 – Пошук закладів та перегляд детальної інформації про заклад харчування

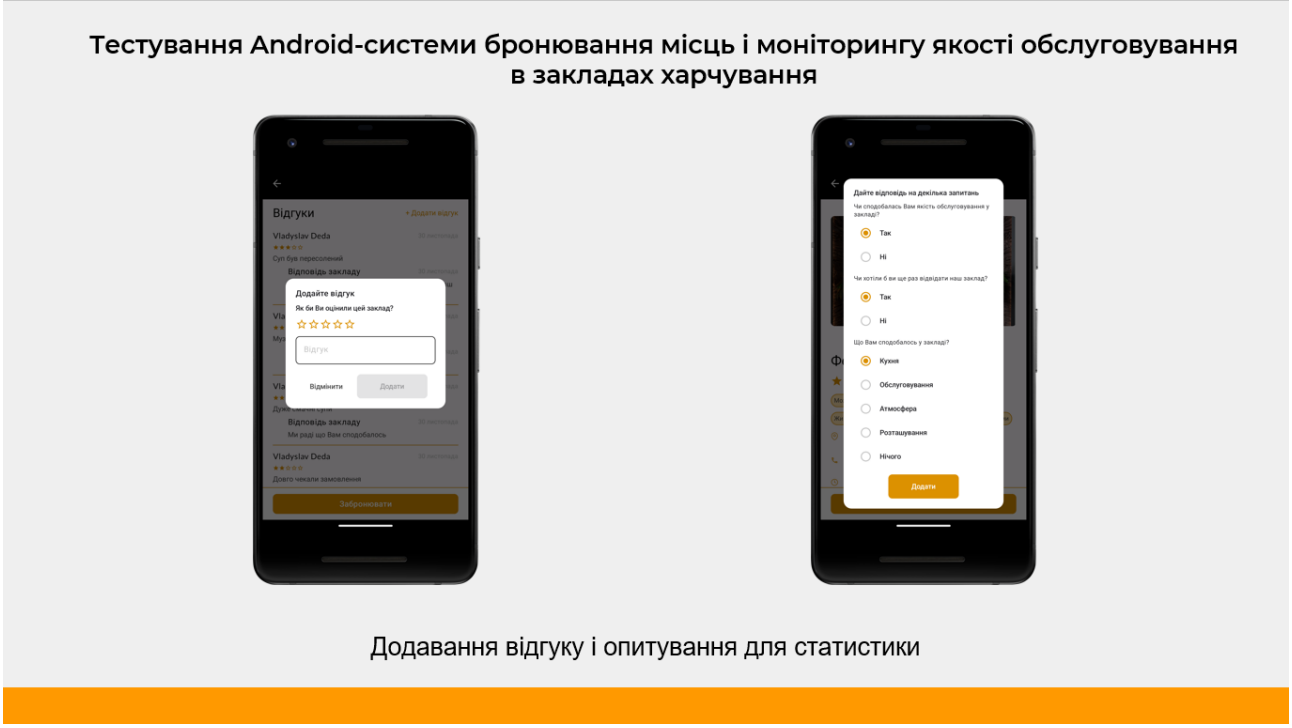



Рисунок Г.12 – Додавання відгуку та опитування для статистики

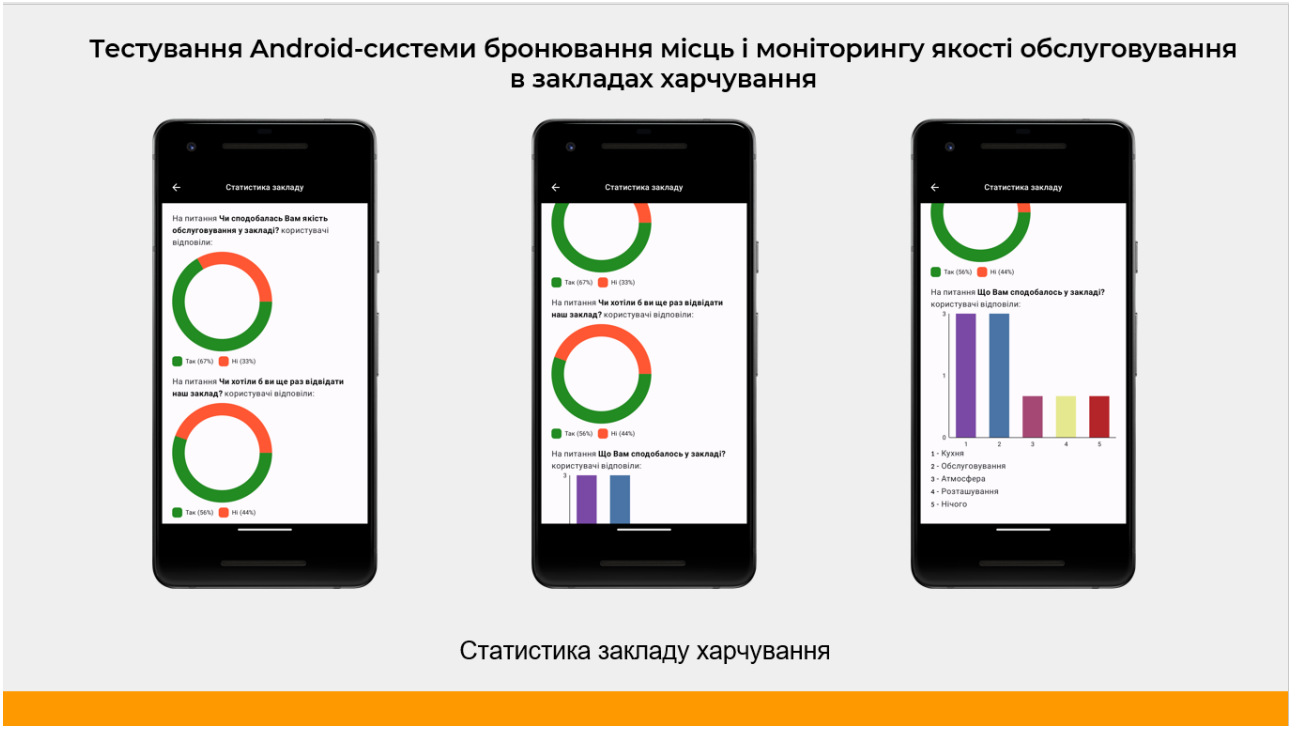



Рисунок Г.13 – Статистика закладу харчування

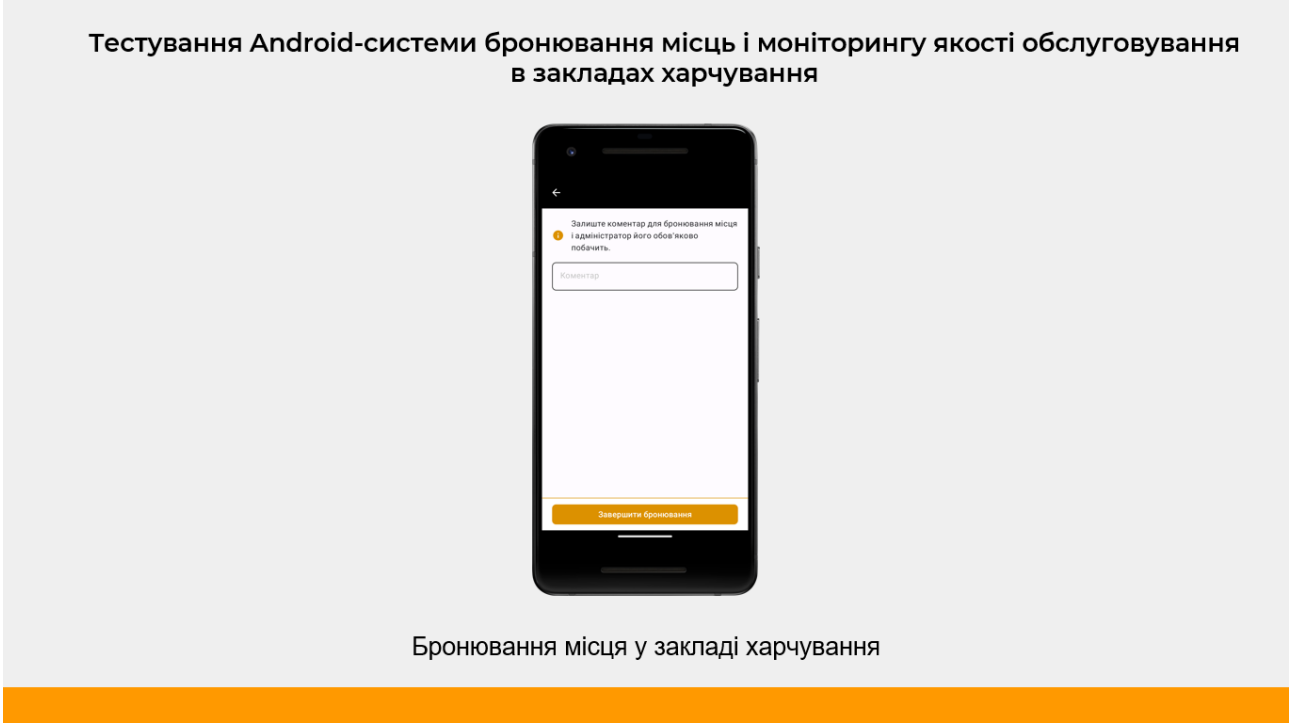



Рисунок Г.14 – Бронювання місця у закладі харчування

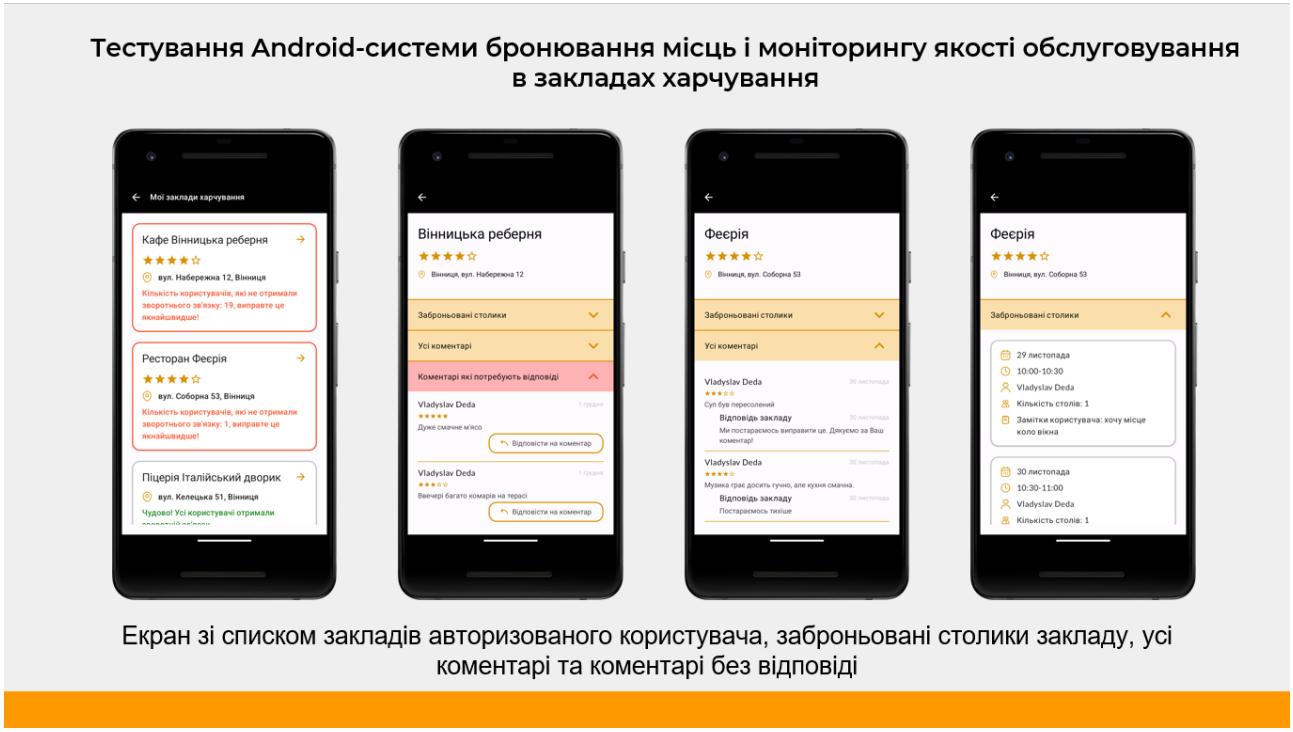



Рисунок Г.15 – Екран зі списком закладів авторизованого користувача, забронювані столики закладу, усі коментарі та коментарі без відповіді

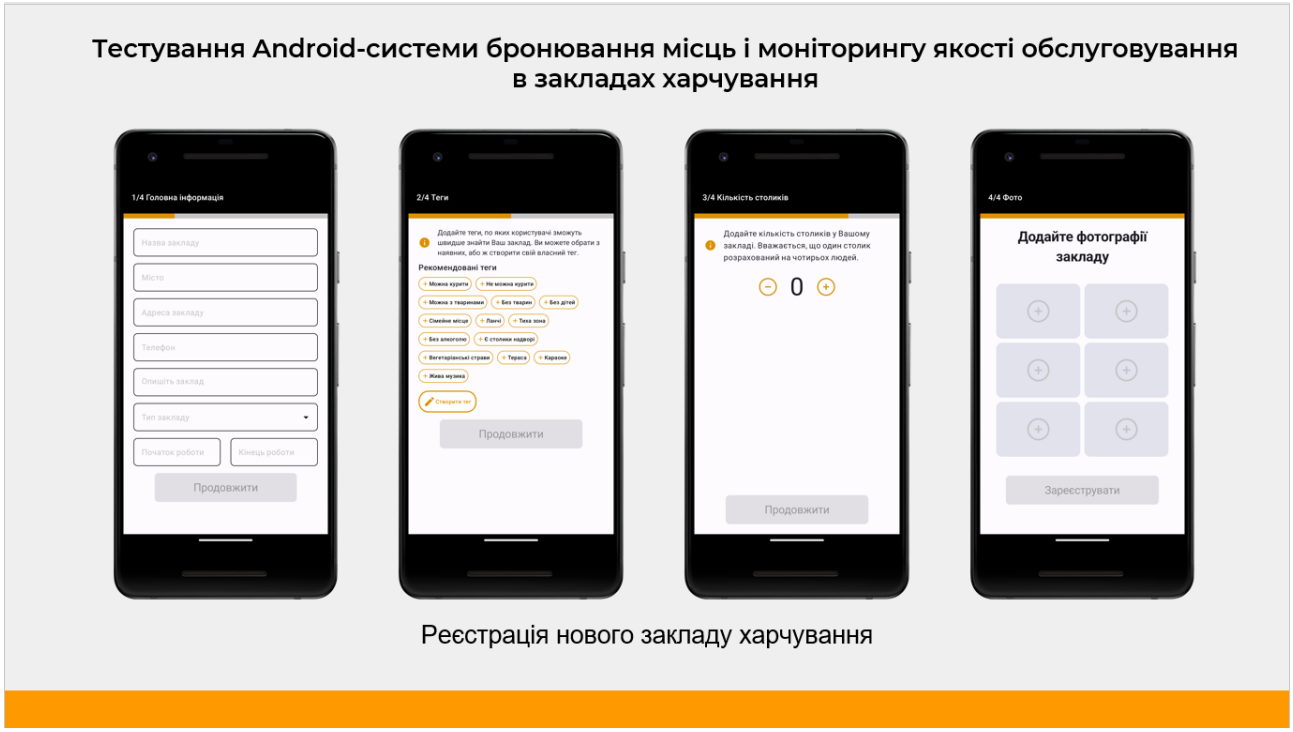



Рисунок Г.16 – Реєстрація нового закладу харчування

## Апробації та публікації

- Апробація матеріалів магістерської кваліфікаційної роботи. Результати магістерської кваліфікаційної роботи обговорювалися на міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання,доступ,2023».

- Публікації. Результати дослідження опубліковані в тезах доповіді міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання,доступ,2023».

Рисунок Г.17 – Апробації та публікації

## Висновки

- під час виконання магістерської кваліфікаційної роботи було розроблено методи та програмні засоби Android-системи бронювання місць і моніторингу якості обслуговування в закладах харчування;
- проведено аналіз поточного стану даного питання, описані основні аналоги, виокремлені їхні особливості та недоліки.
- розроблено методи, моделі та алгоритми роботи Android-системи ;
- реалізовано програмні модулі Android-системи для бронювання місць і моніторингу якості обслуговування в закладах харчування;
- проведено тестування та розроблено інструкцію користувача.

Рисунок Г.18 – Висновки

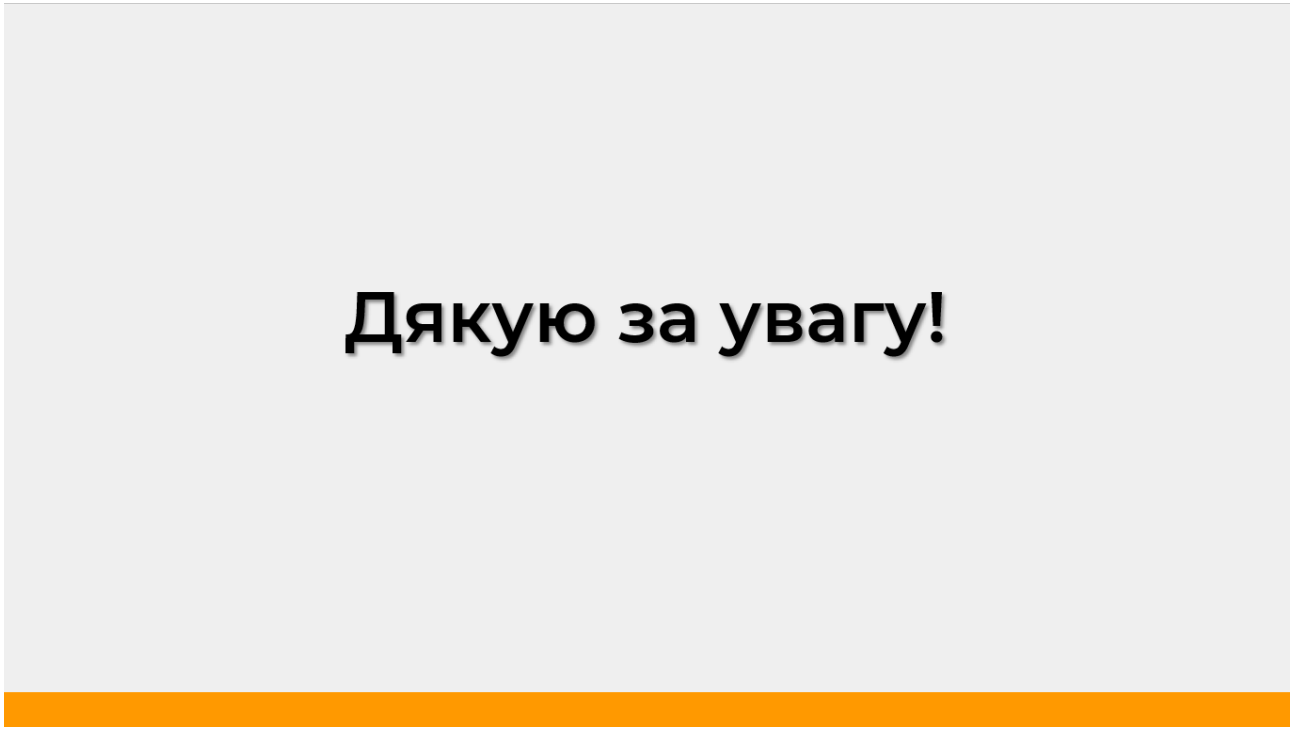



Рисунок Г.19 – Фінальний слайд