IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

BEAUTY SALON

Prepared by:
Anesa Cutuna

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

Date of submission: 21.6.2023

Table of Contents

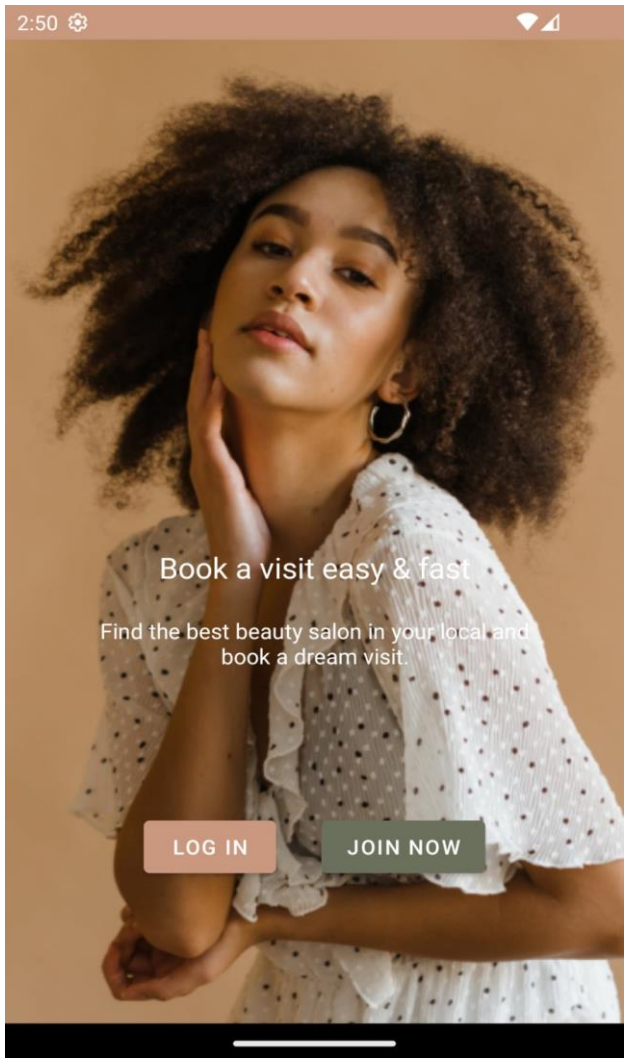
1. Introduction

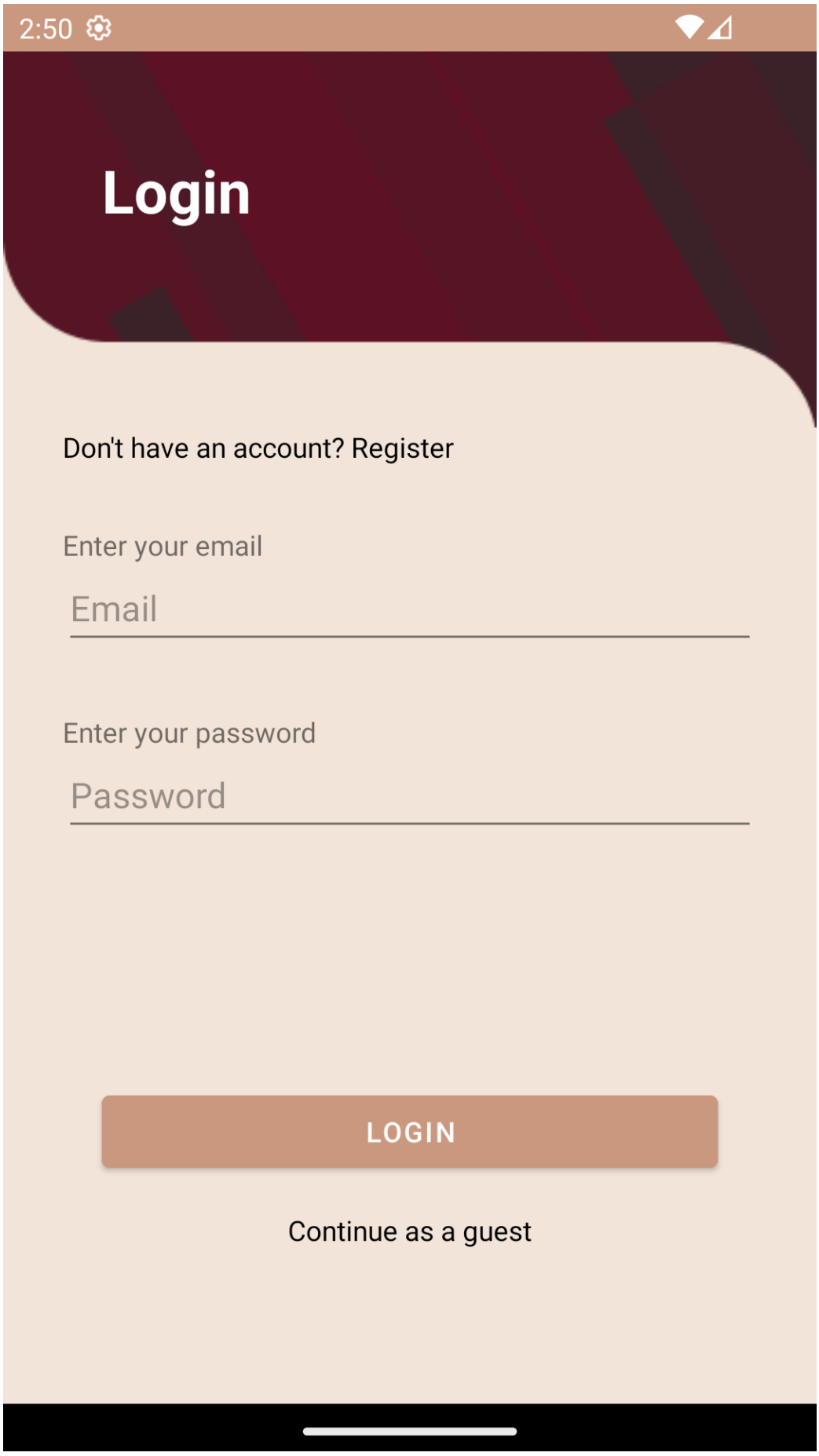
1.1. About the Project

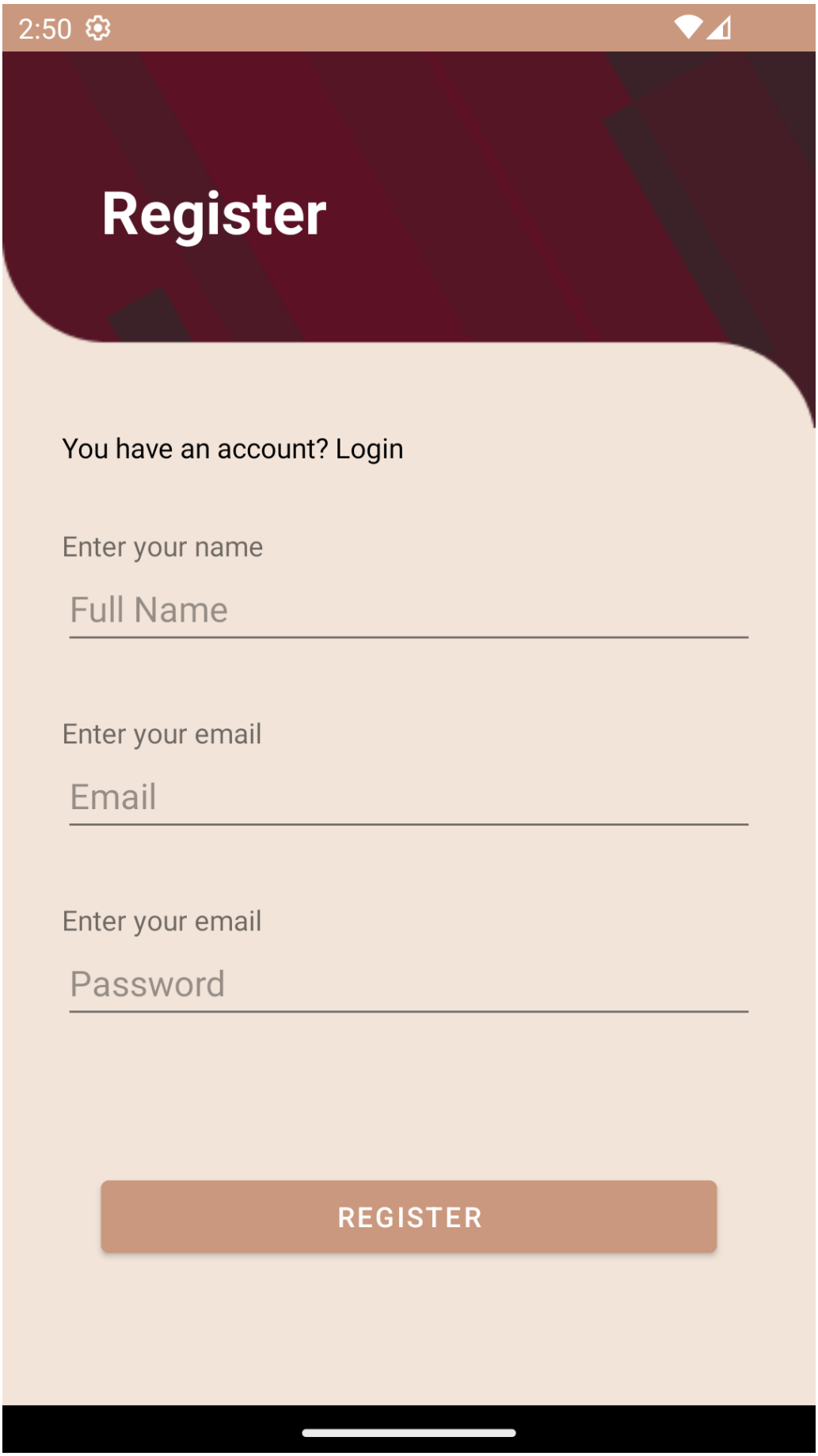
he application is used to digitally manage our studio. BeautyFirst studio offers a range of services. The studio application is used to display the services offered by the beauty studio and to offer online appointment scheduling. It will be possible to display all available services in the studio, along with the corresponding prices. When making an appointment, users are asked to select a desired day and time when they select the service they want. Users can create accounts, log in, make bookings, and check the progress of their appointments via the app . This app should make it easier for potential consumers to find all the services they need in one location and book the treatments they want whenever, wherever and however they want.

This documentation gives a thorough rundown of a Kotlin-based mobile app for booking salon appointments. The application integrates different classes for effective code management and makes use of the Jetpack navigating component for fluid navigating. Additionally, a bespoke API that supports functionality like adding, removing, and modifying salon categories and services has been created using Node.js and Express.

1.2. Project Functionalities and Screenshots




2:50
Login
Don't have an account? Register
Enter your email
Email
Enter your password
Password
LOGIN
Continue as a guest


2:50
Register
You have an account? Login
Enter your name
Full Name
Enter your email
Email
Enter your email
Password
REGISTER

2:51

Sarajevo

30

Top services

Hair saloon

Nails

Make up

pranje kose Approved

2

pranje kose Approved

2

2:51 3G

Search

pranje kose Approved

2

pranje kose Approved

2

testa Denied

1

2:52

Create appointment

Enter Title

Enter Description

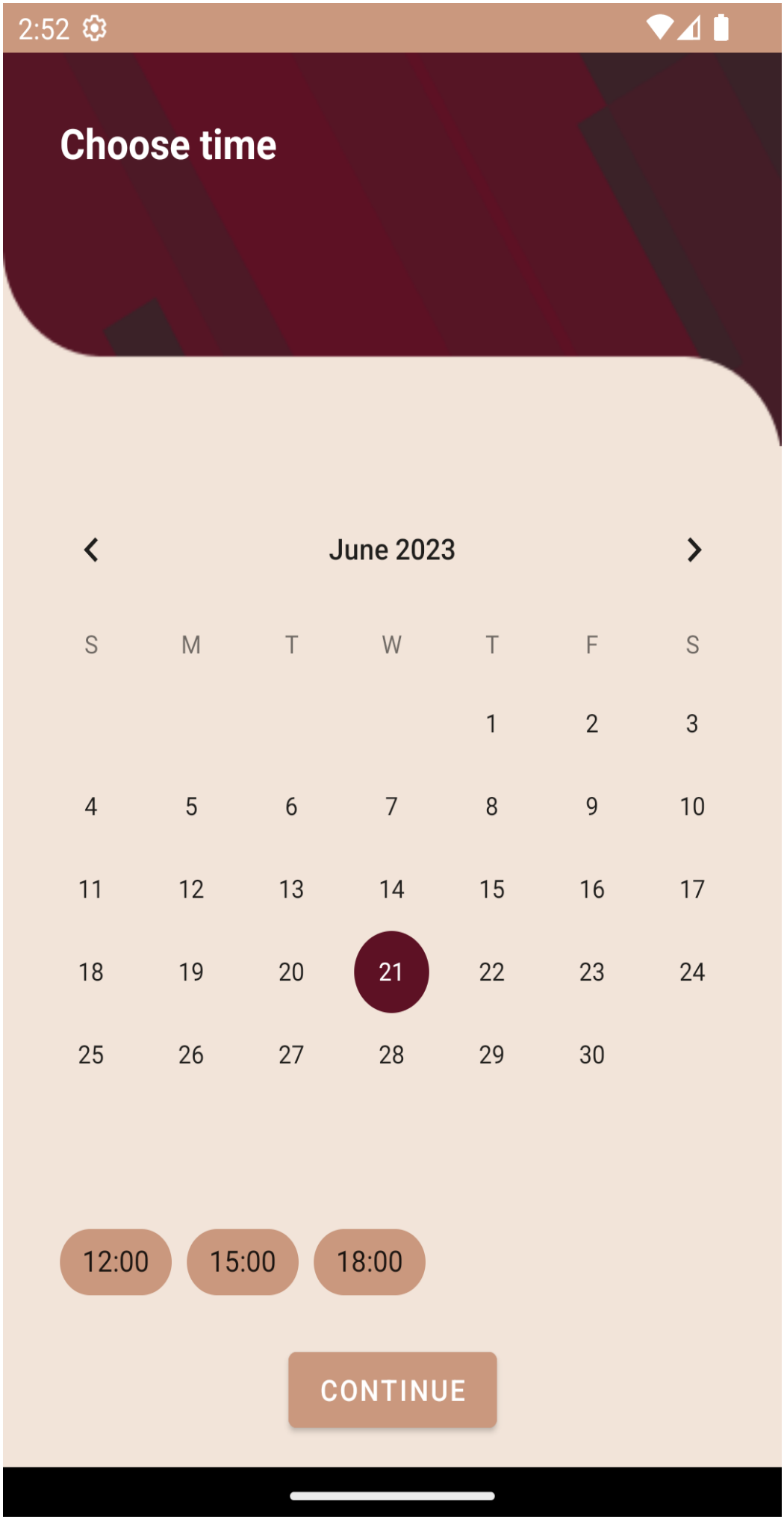
CHOOSE TIME

Hair saloon

Nails

Make up

SUBMIT


2:52
Choose time
June 2023
S M T W T F S
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
12:00
15:00
18:00
CONTINUE

2:52

Profile

LOG OUT

List of the main features:

- Register
- Home Screen and Services
- Online appointment scheduling
- Booking Management
- Service availability
- User account.

2. Project Structure

2.1. Technologies

Kotlin is the primary language for Android app development, offering concise syntax, strong safety features, and Java interoperability. It's ideal for building robust and maintainable Android applications. The app uses the Jetpack Navigation component for easy management of navigation graphs, destinations, and transitions. This enables seamless and intuitive screen navigation. For salon category and service management, the app utilizes a custom API built with *Node.js* and *Express*. This ensures scalable and efficient handling of salon-related operations. Critical data such as user profiles and bookings is stored in a structured database, guaranteeing data integrity and facilitating smooth user experiences and accurate booking management.

2.2. Database Entities

bookings: Stores information about bookings made by users. It includes fields like id (booking ID), title (booking title), description (booking description), serviceId (ID of the service associated with the booking), published (flag indicating if the booking is published), userId (ID of the user who made the booking), and status (booking status).

services: Contains information about different services offered. It includes fields such as id (service ID), title (service title), and description (service description).

users: Stores user information. It includes fields like id (user ID), fullName (user's full name), email (user's email address), and password (user's password).

These tables/entities represent the core components of the database/schema for the given application.

2.3. Design Patterns

In my project, I have implemented a layered architecture approach to ensure clear separation of responsibilities and improve modularity. The components are divided into different layers, each with its specific role and task.

Firstly, I have the Adapters layer, which is responsible for handling the RecyclerViews. Adapters are crucial in binding data to the views and providing the necessary functionality for displaying and interacting with the data in a RecyclerView.

Next, I have the NetworkService layer, which takes care of networking-related tasks. This layer handles communication with remote servers, manages API calls, and processes the received data. By having a separate layer for network operations, I can ensure that network-related code is encapsulated and easily maintainable.

Lastly, I have the Fragments layer, which is responsible for managing the layout and business logic of the application. Fragments provide the visual representation of the user interface and handle user interactions. They encapsulate the presentation layer and contain the necessary logic for displaying and manipulating data.

By dividing my project into these layers, I can achieve a clear separation of concerns and promote code reusability and maintainability. Each layer has its specific role, allowing for easier testing, debugging, and future enhancements.

2.4. Tests

```
//test if user id is saved and user is registered
fun testIsUserRegistered() {
    //given
    val userId = 0

    //when
    val result = Utils().isUserRegistered(userId)

    //then
    Assert.assertEquals( expected: false, result)
}
```

```kotlin
//test if user id is saved and user is registered
fun testIsUserRegisteredWhenUserIdNull() {
    //given
    val userId = null

    //when
    val result = Utils().isUserRegistered(userId)

    //then
    Assert.assertEquals( expected: false, result)
}
```

```kotlin
//test if user id is saved and user is registered
fun testIsUserRegisteredWhenUserIdIsValue() {
    //given
    val userId = 3

    //when
    val result = Utils().isUserRegistered(userId)

    //then
    Assert.assertEquals( expected: true, result)
}
```

```kotlin
//test if api response is successfull when it is successful
fun testIsApiRespnseSuccessfulIfItIs() {
    //given
    val response: Response<Any> = Response.success( body: 200)

    //when
    val result = Utils().isApiCallSuccessful(response)

    //then
    assertEquals( expected: true, result)
}
```

3. Conclusion

Users may book salon services quickly and easily with the help of this mobile Kotlin-developed application. A seamless user experience is guaranteed by the use of the Jetpack Navigation component and a well-organized codebase. Effective management of salon categories and services is made possible by the custom API, which was created with Node.js and Express. This program has the potential to develop into a complete and user-friendly solution for salon booking requirements with more improvements in the future.