**MAHATMA GANDHI MEMORIAL COLLEGE**

**UDUPI -576102**



**PROJECT REPORT 2022-23**

**“SKIN SAVIOUR ANDROID APPLICATION”**

**DEVELOPED BY**

**Nishan Hedge Reg No:201161522130**

**Yathish Reg No:201161522196**

**Rohan Reg No:201161522204**

**Under the Guidance of**

**“Prof. Mr. Rajamoorthi Rao”**

**Dept. of Computer Science**

**Submitted to the Mangalore University in partial fulfilment of the Award of**

**Bachelor in Computer Application**

**MANGALORE UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE**

**MAHATMA GANDHI MEMORIAL COLLEGE**

**UDUPI-576102**

# INDEX

|  |  |  |
| --- | --- | --- |
| CHAPTER | DESCRIPTION | PAGE NO |
| 1. | Project Title | 3 |
| 2. | Introduction | 3 |
| 3. | Synopsis  3.1 Title of the project  3.2 Introduction and objectives  3.3 Project Category  3.4 Tools/Platform  3.4.1 Hardware/Software Requirements  3.5 Modules  3.5.1 Module Used  3.5.2 Module Description | 3-9 |
| 4. | Tools and Environment  4.1 Front End  4.2 Back End | 9-11 |
| 5. | Overall Description  5.1 Product Perspective  5.2 Product Function  5.3 User Character  5.4 General Constraints | 11-12 |
| 6. | Software Requirements and Specification  6.1 Analysis  6.1.1 CFD’s  6.1.2 DFD’s | 12-17 |
|  | 6.1.3 Data Dictionary |  |
| 7. | Structure of System Design  7.1 Database Diagram  7.2 Table Structure | 17-21 |
| 8. | Program Code Listing | 22-95 |
| 9. | User Interface (Screens and Reports)  9.1 Login  9.2 Main  Screen/Home Page  9.3 Menu  9.4 Data Store/Retrieval/Update  9.5 Validation  9.6 Error Message | 96-107 |
| 10. | Testing  10.1 Introduction  10.2 Test Reports  10.2.1 Unit Testing  10.2.2 Integrate Testing  10.2.3 System Testing | 108-117 |
| 11. | Conclusion | 118 |
| 12. | Limitations | 118 |
| 13. | Scope for enhancements | 118 |
| 14. | Abbreviations and Acronyms | 118-119 |
| 15. | Bibliography | 119 |

**3.1 Title of the Project**

Scissors Sound

Android Application

**3.2 Introduction**

"Welcome to our Men's Salon Booking App! With our app, you can easily book appointments at our salon and enjoy a hassle-free grooming experience. Whether you're in need of a stylish haircut, a fresh shave, or any other grooming service, our app has got you covered.

For salon owners, our app provides a convenient way to track and manage appointments. You'll have full visibility into the available time slots and be able to efficiently manage your salon's schedule. Say goodbye to manual booking systems and embrace the digital revolution with our user-friendly app.

As a user, you'll find it incredibly easy to book appointments. Simply browse through the available time slots and select the one that fits your schedule. Whether you prefer an early morning appointment or a late evening slot, our app offers flexibility to accommodate your needs. You can also explore our range of services, view our skilled stylists, and make your selection with confidence.

We understand the importance of punctuality and value your time. With our app, you can be assured that your appointment will be honored, allowing you to plan your day accordingly.

**Objective**

The objective of our Men's Salon Booking App is to provide a seamless and efficient platform for both salon owners and customers to manage and book appointments.

For salon owners, our objective is to streamline their appointment management process, replacing traditional manual systems with a digital solution. By offering an easy-to-use interface, our app aims to simplify the tracking and organization of appointments, allowing salon owners to optimize their scheduling and resources. We strive to empower salon owners with tools that enhance their efficiency and improve the overall salon experience for their customers.

**Objectivesof the project**

* Main Objective of this App is Computerize all the work and update details in Database.
* To provide convenient and user-friendly platform for booking appointments at their preferred time slots.
* To Effectively manage the System.
* To maintain Appointment details and schedule.
* To generate Bill.
* Better Customer Service.

**Scope**

* It encompasses various features and functionalities to provide a seamless and satisfying customer experience.
* Customer can benefit from reading reviews and ratings.
* Accessible from anywhere in the world, expanding the market reaches.
* It encompasses the development and implementation of a comprehensive mobile application that facilitates appointment scheduling and management for a men’s salon.
  1. **Project Category**
* Android Application

**3.4 Hardware/Software Requirement**

**Android versions specifications:**

* Android pie or more

**Hardware Requirements:**

* Operating system : Windows 10 or higher version
* CPU : Intel core i3 or higher version
* RAM : 4GB RAM
* Processor Speed : 3.00 GHz

**Other Hardware :**

* Android Phone

**3.5 MODULES**

**3.5.1 Number of Modules Used**

**Module 1 : User**

1.1-Splash Screen

1.2-Sign Up

1.3-Login

1.4-Forgot Password

1.5-Home

1.6-Book Appointment

1.7-Scheduled List

1.-Payment

**Module 2 : Admin**

2.1-Splash Screen

2.2-Login

2.3-Delete order

**3.5.2 Module Descriptions**

**Module 1 : User**

**1.1-Splash Screen**

When you click it Scissors Sound will initially show splash screen with logo after 3 seconds it will move to login screen.

**1.2-Sign Up**

Sign up is an action to register users for Scissor Sound app it create a new user account to user though Username, Email and Password.

**1.3- Login**

In this login page user by providing proper username and password user can successfully login to user account.

**1.4-Forgot Password**

If suppose user forgot his password then by clicking on forgot password, he will able to change his password.

**1.5-Home**

This module contains all salon stylist name, photo and review of stylists and button to next page that have salon categories like haircut, shaving and oil treatment other category its depends on stylist.

**1.6- Book Appointment**

Appointment page contains hair stylist’s service list with book now button to appoint according to time and date.

**1.7-Scheduled List**

Its page containing all service request from costumer with name of stylist, service details and date & time.

**1.8-Payment**

Payment can be done through online payments and cash on delivery options user can select any one of this options for payment.

**Module 2 : Admin**

**2.1-Splash Screen**

When you click it Scissors Sound app will initially show splash screen with logo after 3 seconds it will move to login screen.

**2.2-Login**

Admin page login contains a specific email address and password is this specific login page admin can only access.

**2.3-Delete Order**

Contains all request of service from customers with their name, phone number, date & time, service details, price with delete order button to delete service order for admin.

**Type of The Project**

-User Defined Project

**4. Tools And Environment**

**4.1 Front End**

**About Flutter**

* Flutter is a powerful open-source UI framework developed by Google for building natively compiled applications for multiple platforms, including Android, iOS, web, and desktop from a single codebase. This enables developers to write code once and deploy it across various platforms, saving time and effort
* Flutter uses Dart as its primary programming language. Dart is a modern and object-oriented language with a syntax similar to Java and JavaScript. It is easy to learn and provides features like hot-reload, which allows developers to see the changes instantly without rebuilding the entire app.
* Flutter provides an extensive collection of customizable and reusable widgets for building beautiful and responsive user interfaces.
* Flutter apps are known for their fast and smooth performance. Flutter utilizes a rendering engine called **Skia,** which directly renders the UI elements without relying on native controls. This allows Flutter apps to achieve high performance and deliver a native-like experience.
* Flutter provides a suite of tools for testing and debugging applications. It includes a powerful testing framework for unit, integration, and widget testing. Additionally, the Flutter DevTools offer a set of debugging and performance analysis tools to help diagnose and optimize the app's performance.

**4.2 Back End**

**About Google Firebase**

* Google firebase is a Google backend application development software that enables developers to develop IOS, Android and Web apps. Firebase provides tool for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.
* Google Analytics for firebase offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behaviour in IOS and android apps, enabling better decision making about improving performance and app marketing.
* Firebase authentication makes it easy for developers to build secure authentication system and enhances the sign in and on boarding experiences for users. This feature offers a complete identity solution supporting email and password accounts, phone auth as well as Google, Facebook, GitHub, Twitter login and more.
* Firebase storage provides a simple way to save binary files most often images, but it could be anything to Google cloud storage directly from the client.
* Firebase sends you new data as soon as it’s updated when your client saves a change to the data, all connected clients receive the updated data almost instantly.

**5.OVERALL DESCRIPTION**

**5.1 Product Perspective**

Product of this project is the Men's Salon Automation Application, which helps the customer to appoint the salon service at home though Internet. This application efficient in managing service to customer and remove waiting time of customers.

**5.2 Product Functions**

* The app provides user friendly interface, which is convenient to use.
* This app provide efficiency in managing salon shop.

**5.3 User Character**

The user of the system are customer of Men’s salon. Customer has to register to the app with email id and password.

**5.4 General Constraints**

App should run on any android mobile which has android version more than 9.

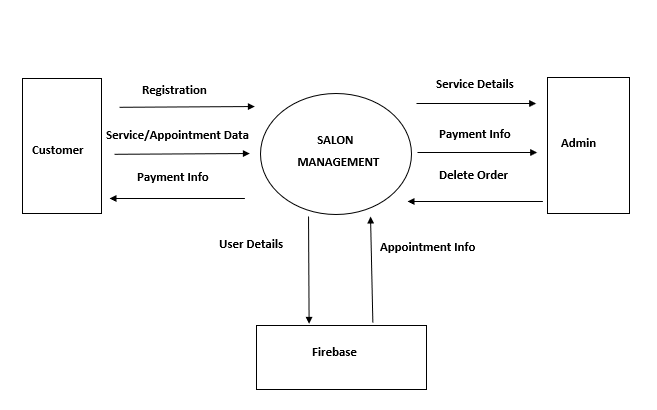
**6. SOFTWARE REQUIREMENTS AND SPECIFICATION**

A Software Requirement Specification (SRS) is the complete description about what the software will do and how the software is expected to perform it. It also describes the functionality the software needs to fulfil all the user needs.it is a formal report that enables the users review whether SRS is according to their requirements.

**6.1 Analysis**

**6.1.1. Context Flow Diagram**

* Context flow diagram is the graphical representation of the flow of the context through an information system.
* CFD’s can also be used for the visualization of the data processing on a CFD ,data item flow from an external data source or internal data store or external data sink via an internal process.
* A CFD provides no information about the timing or ordering of the processes or about whether process will operate in sequence or in parallel.
* CFD is quite different from a flow chart which shows the control through an algorithm, allowing a reader to determine what operations will be performed in what order and under what circumstances, but not what kind of data will be input to and output from the system nor where the data will come from and go nor where the data will be stored.



**6.1.2 Data Flow Diagram**

The data flow diagram describes the flow of data with the help of various in a clear way.

The DFD server two purposes:

* To provide an indication of flow of data are transformed as they move through the system.
* To depict the function that transforms the data flow.
* It provides additional information that is used during the analysis of the information domain and server as a basis for the modeling function. The DFD is also known as a Data Flow Graph or Bubble chart.

The DFD may be used to represent a system or software at any level of abstraction. In fact DFD may be partitioned into levels that represent increasing information flow and functions details. Therefore modeling as well as information flow modeling.

A level 0 DFD also called a fundamental system module or a contact module represents the entire software elements as single bubble with input and output data indicated incoming and outgoing arrows respectively.

Additional process and information flow paths are presented as the levels 0 DFD is partitioned to reveal more details. For e.g.: level 1 DFD might contain four or five bubbles with interconnecting arrows. Each of the processes represented at level 1 is sub function of the overall system depicted in the context model

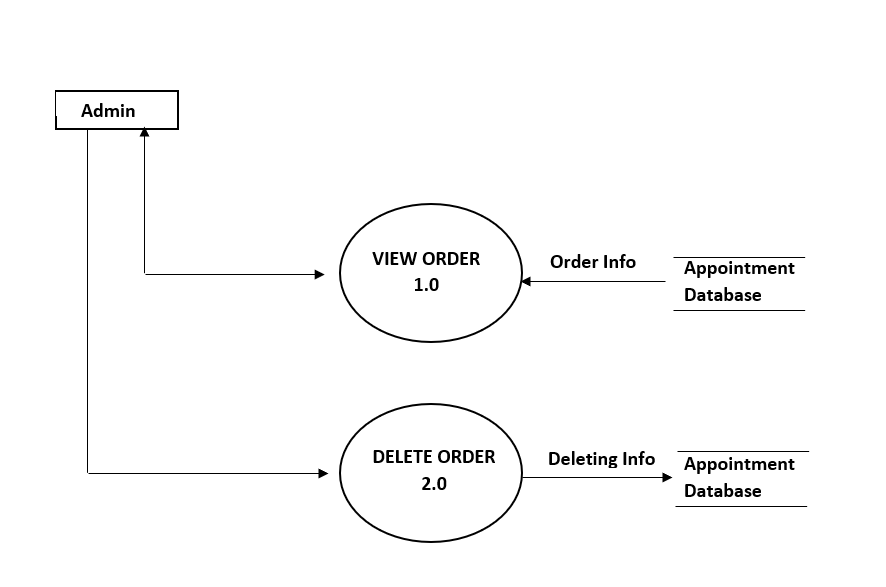
The base notation used to create a DFD makes easy to analyze understand the DFD is graphical tool that can be very valuable during software requirement analysis .

**NOTATION FOR DFD DIAGRAM**

NOTATION DISCRIPTION

|  |  |
| --- | --- |
|  | A circle represents a process applied to data or control or changes if in same way. |
|  | A rectangle is used to represent an external Entity i.e. a system element or another System that produce information by the Software or receiver information produced By the software. |
|  | An arrow represent one or more data items or data objects |
|  | The open box represent data stored Information that is used by the software |

**DFD**



**Customer**

**Login**

**Database**

**Register/Login**

**REGISTRATION**

**1.0**

**Register/Login**

**Booking Details**

**Confirmation Info**

**Appointment Info**

**Appointment**

**Database**

**Retrieving Info**

**Payment Info**

**Confirmation Info**

**Payment**

**Database**

**BOOKED**

**DETAILS**

**3.0**

**2.0a**

**BOOK APPOINTMET**

**Book Appointment**

**Booking Info**

**Order Summery**

**PAYMENTS**

**4.0**

**6.1.3 Data Dictionary**

1. Login @Email + Password

Email: [a-zA-Z0-9.\_-]+@[a-z]+\\.+[a-z]+

Password: ^(?=.\*[A-Z])(?=.\*[a-z])(?=.\*[0 9])

(?=.\*[<>/+@#$])(?=\\S+$).{8,}$

2. Signup: Username+@Email +Password

Username: :^(?=.\*[A-Z])(?=.\*[a-z])(?=.\*[0 9])

Email: [a-zA-Z0-9.\_-]+@[a-z]+\\.+[a-z]+

Password: ^(?=.\*[A-Z])(?=.\*[a-z])(?=.\*[0 9])

(?=.\*[<>/+@#$])(?=\\S+$).{8,}$

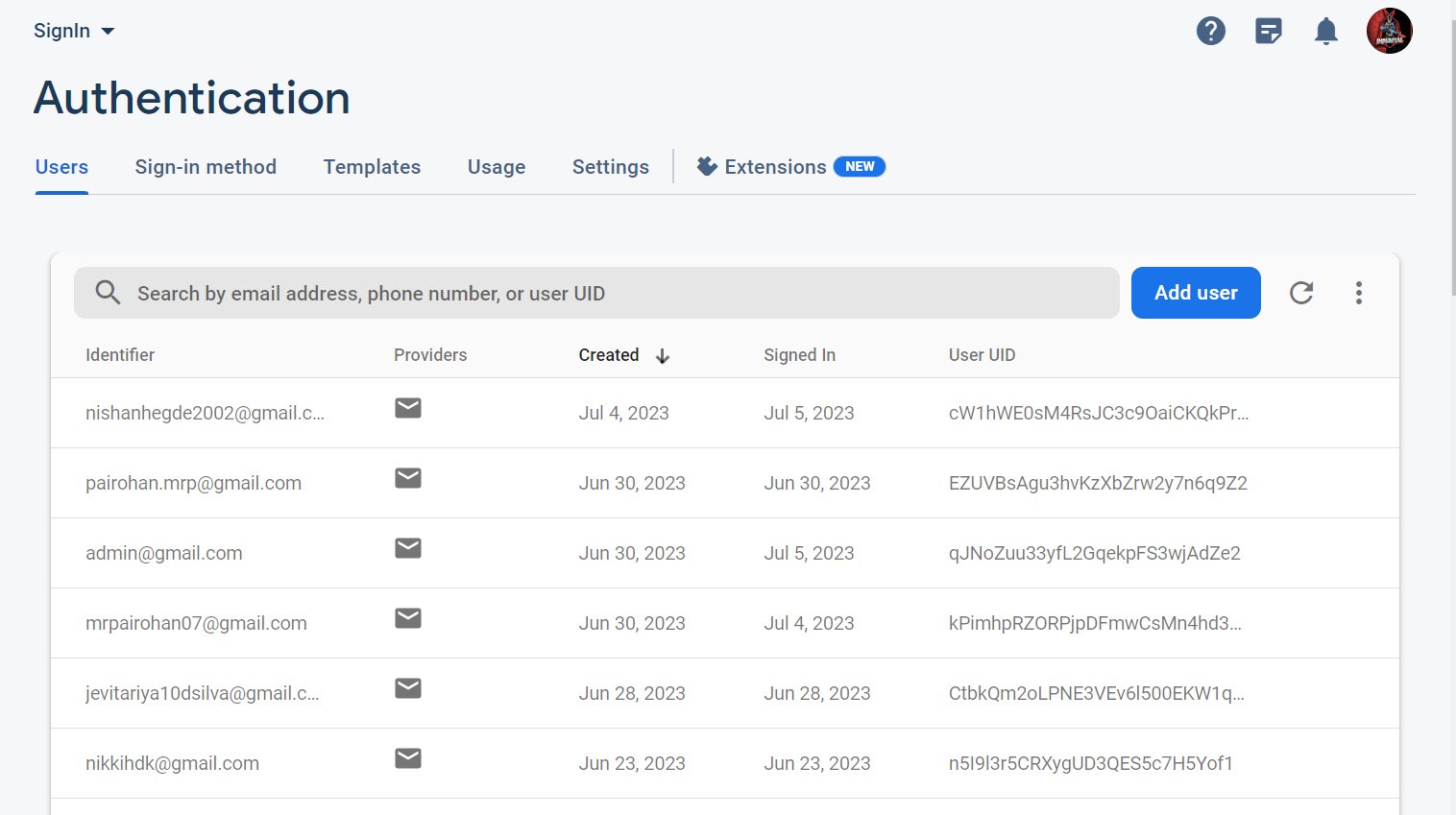
**7. STRUCTURE OF SYSTEM DESIGN**

**7.1 Database Diagram**

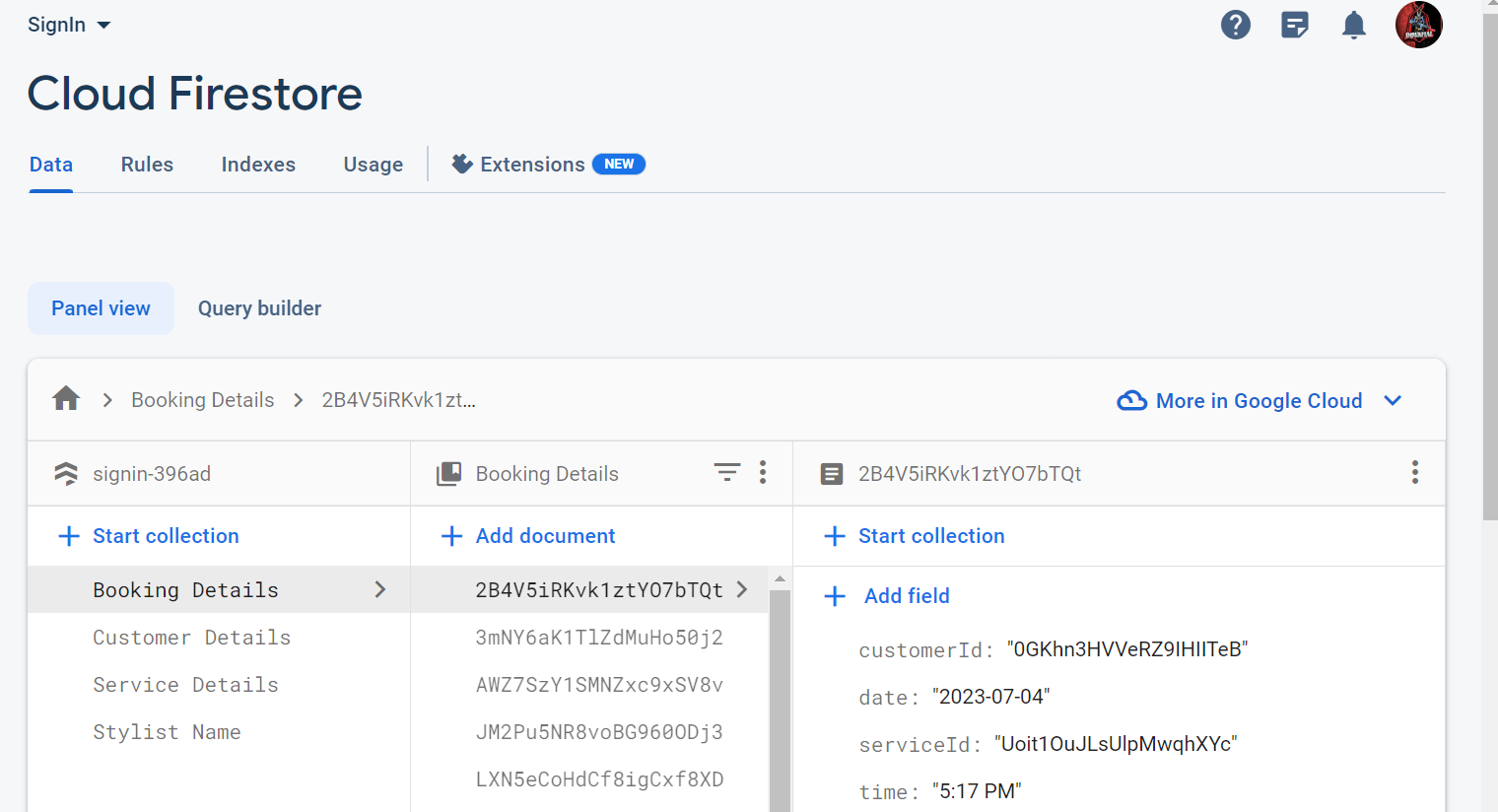
Database Design(Firebase Firestore Database)

Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects.

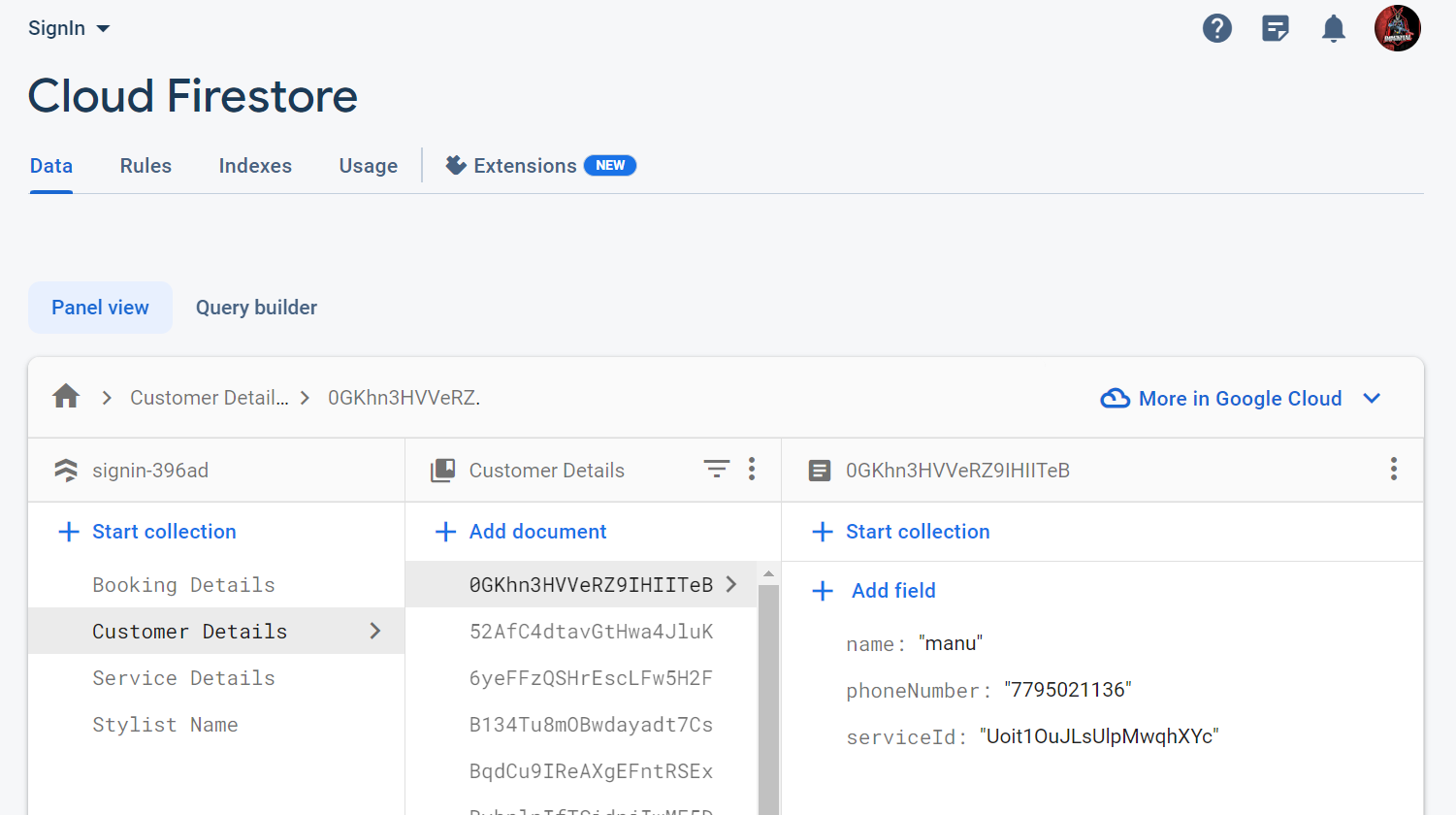
**Users Login**



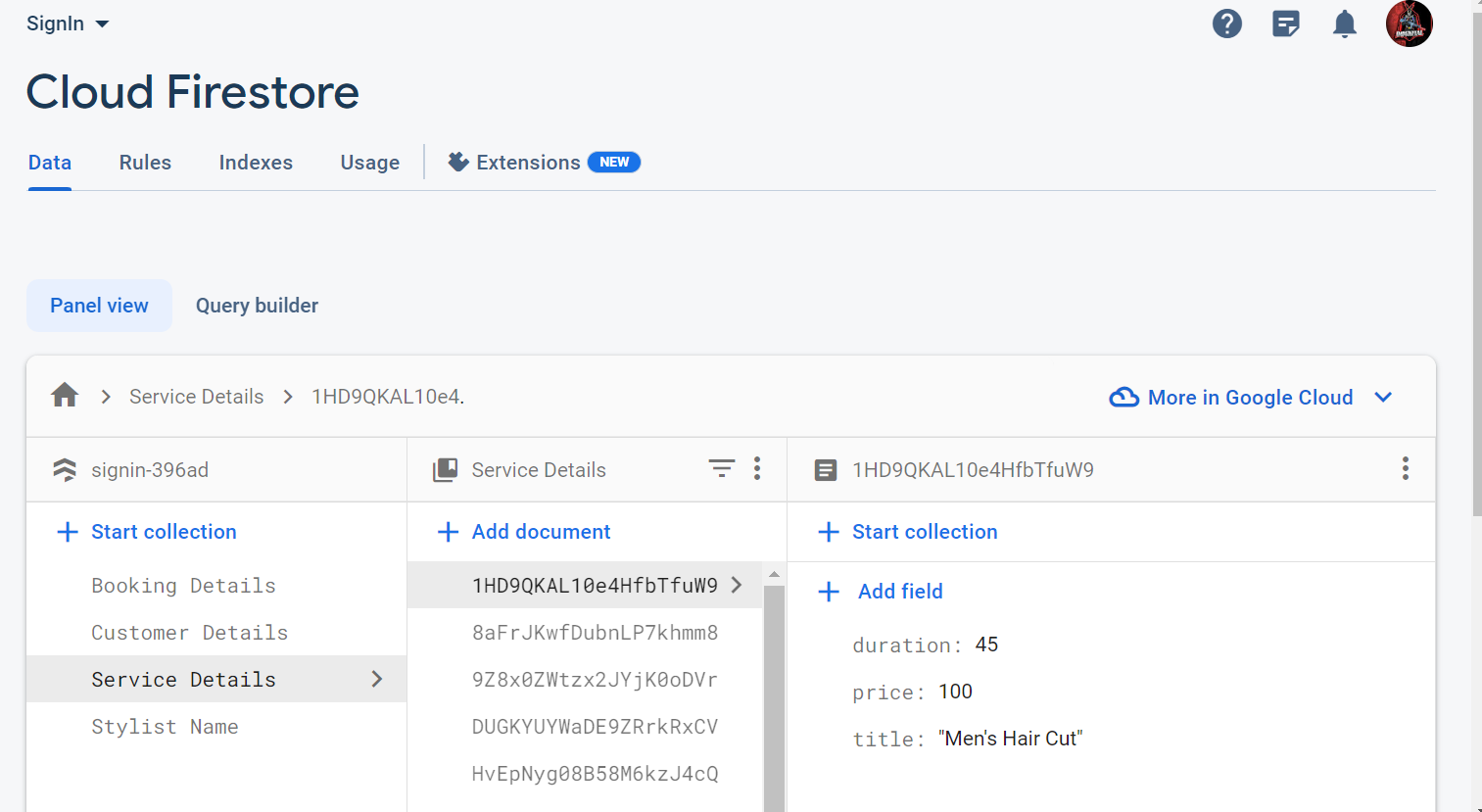
**Booking Details**



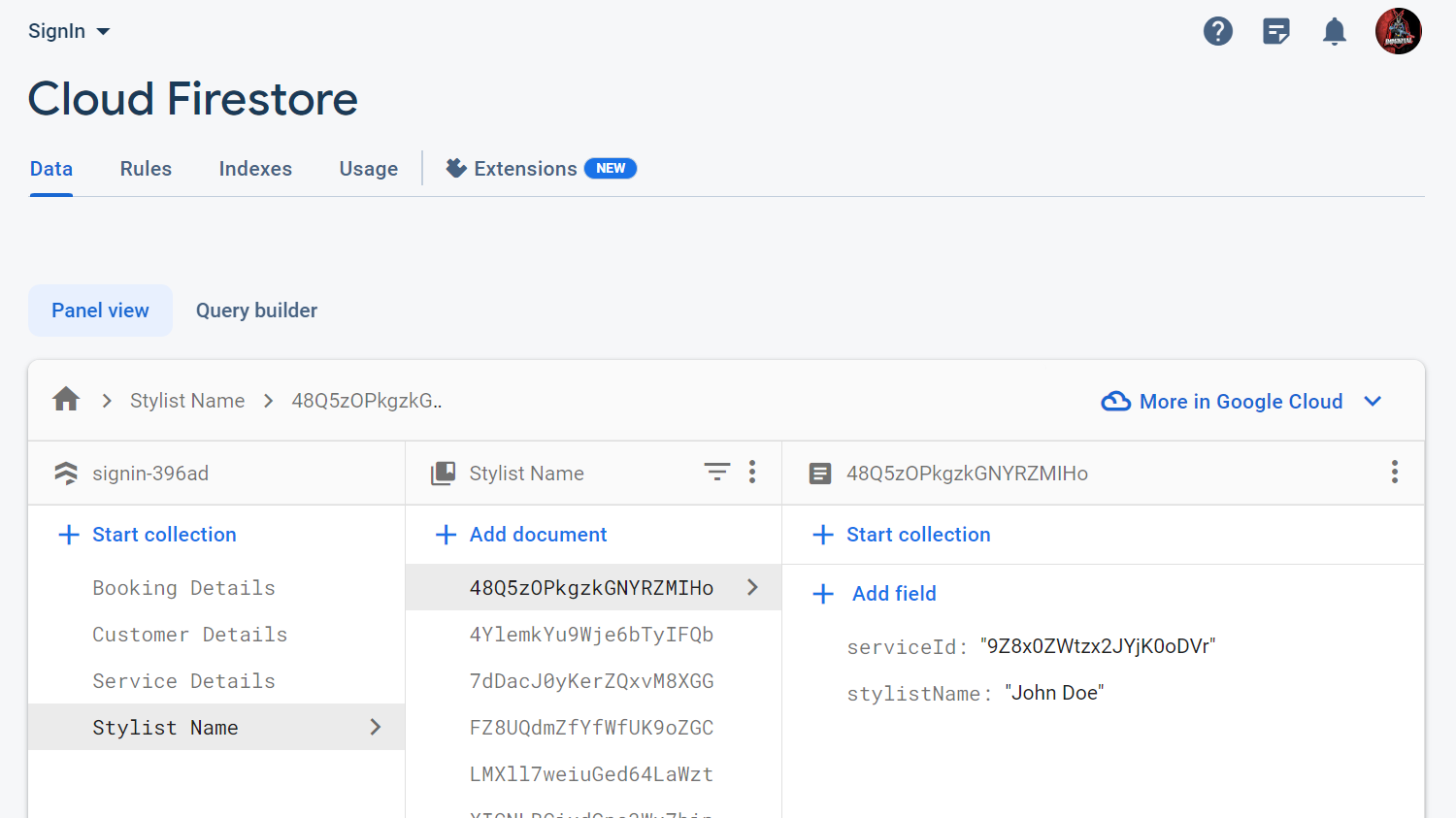
**Customer Details**

****

**Service Details**

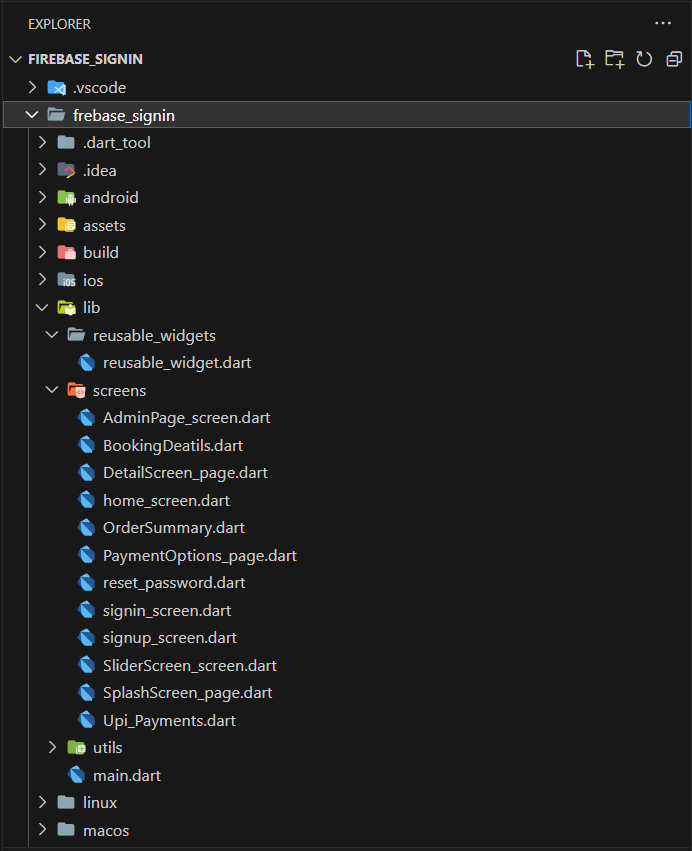
****

**Stylist Name**



#### **8. Program Code Listing**

##### **Project Structure**



###### **DART CODE:**

**MAIN.DART**

import 'package:firebase\_core/firebase\_core.dart';

import 'package:flutter/material.dart';

import 'package:frebase\_signin/screens/SplashScreen\_page.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'screens/signin\_screen.dart';

import 'screens/home\_screen.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp();

  SharedPreferences prefs = await SharedPreferences.getInstance();

  bool isLoggedIn = prefs.getBool('isLoggedIn') ?? false;

  runApp(MyApp(isLoggedIn: isLoggedIn));

}

class MyApp extends StatefulWidget {

  const MyApp({Key? key, required this.isLoggedIn}) : super(key: key);

  final bool isLoggedIn;

  @override

  \_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

  late String initialRoute;

  @override

  void initState() {

    super.initState();

    initialRoute = '/splash'; // Set the splash screen as the initial route

    \_navigateToNextScreen();

  }

  void \_navigateToNextScreen() async {

    await Future.delayed(Duration(seconds: 3)); // Add a delay for the splash screen

    setState(() {

      initialRoute = widget.isLoggedIn

          ? '/home'

          : '/signin'; // Navigate to the appropriate screen

    });

  }

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(

        // Your theme data here

      ),

      initialRoute: initialRoute,

      routes: {

        '/splash': (context) => SplashScreen(), // Add the splash screen route

        '/signin': (context) => SignInScreen(),

        '/home': (context) => HomeScreen(username: ''),

      },

    );

  }

}

**SIGNIN\_SCREEN.DART**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:frebase\_signin/reusable\_widgets/reusable\_widget.dart';

import 'package:frebase\_signin/screens/AdminPage\_screen.dart';

import 'package:frebase\_signin/screens/home\_screen.dart';

import 'package:frebase\_signin/screens/reset\_password.dart';

import 'package:frebase\_signin/screens/signup\_screen.dart';

import 'package:email\_validator/email\_validator.dart';

class SignInScreen extends StatefulWidget {

  const SignInScreen({Key? key}) : super(key: key);

  @override

  \_SignInScreenState createState() => \_SignInScreenState();

}

class \_SignInScreenState extends State<SignInScreen> {

  TextEditingController \_passwordTextController = TextEditingController();

  TextEditingController \_emailTextController = TextEditingController();

  bool \_isPasswordValid = true;

  bool \_isPasswordVisible = false; // Track the password visibility

  @override

  Widget build(BuildContext context) {

    return WillPopScope(

      onWillPop: () async {

        // closing app

        if (FirebaseAuth.instance.currentUser == null) {

          return true;

        } else {

          return false;

        }

      },

      child: Scaffold(

        body: Container(

          width: MediaQuery.of(context).size.width,

          height: MediaQuery.of(context).size.height,

          decoration: BoxDecoration(

            gradient: LinearGradient(

              colors: [

                Color.fromARGB(255, 244, 192, 140), // Dark brown

                Color.fromARGB(255, 242, 189, 136), // Medium brown

                Color(0xFFCDBA96), // Light brown

              ],

              begin: Alignment.topCenter,

              end: Alignment.bottomCenter,

            ),

          ),

          child: SingleChildScrollView(

            child: Padding(

              padding: EdgeInsets.fromLTRB(

                20,

                MediaQuery.of(context).size.height \* 0.2,

                20,

                0,

              ),

              child: Column(

                children: <Widget>[

                  logoWidget("assets/images/Hotpot.png"),

                  const SizedBox(

                    height: 30,

                  ),

                  TextFormField(

                    controller: \_emailTextController,

                    keyboardType: TextInputType.emailAddress,

                    decoration: InputDecoration(

                      labelText: "Enter Email",

                      prefixIcon: Icon(Icons.person\_outline),

                      filled: true,

                      fillColor: Colors.white,

                      border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10.0),

                      ),

                    ),

                  ),

                  const SizedBox(

                    height: 30,

                  ),

                  TextFormField(

                    controller: \_passwordTextController,

                    obscureText:

                        !\_isPasswordVisible, // Toggle password visibility

                    decoration: InputDecoration(

                      labelText: "Enter Password",

                      prefixIcon: Icon(Icons.lock\_outline),

                      suffixIcon: GestureDetector(

                        onTap: () {

                          setState(() {

                            \_isPasswordVisible =

                                !\_isPasswordVisible; // Toggle password visibility

                          });

                        },

                        child: Icon(

                          \_isPasswordVisible

                              ? Icons.visibility

                              : Icons.visibility\_off,

                        ),

                      ),

                      filled: true,

                      fillColor: Colors.white,

                      border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10.0),

                      ),

                    ),

                    validator: (value) {

                      setState(() {

                        \_isPasswordValid = \_validatePassword(value!);

                      });

                      return '';

                    },

                  ),

                  if (!\_isPasswordValid)

                    Text(

                      'Password must contain at least 8 characters, one uppercase letter, one numeric digit, and one special character.',

                      style: TextStyle(color: Colors.red),

                    ),

                  const SizedBox(

                    height: 5,

                  ),

                  forgetPassword(context),

                  firebaseButton(context, "SIGN IN", () async {

                    if (EmailValidator.validate(\_emailTextController.text) &&

                        \_isPasswordValid) {

                      try {

                        UserCredential userCredential = await FirebaseAuth

                            .instance

                            .signInWithEmailAndPassword(

                          email: \_emailTextController.text,

                          password: \_passwordTextController.text,

                        );

                        if (\_emailTextController.text == 'admin@gmail.com' &&

                            \_passwordTextController.text == 'Admin@123') {

                          Navigator.push(

                            context,

                            MaterialPageRoute(

                                builder: (context) => AdminPage()),

                          );

                        } else {

                          //regular user

                          Navigator.push(

                            context,

                            MaterialPageRoute(

                              builder: (context) => HomeScreen(

                                username:

                                    userCredential.user!.displayName ?? '',

                              ),

                            ),

                          );

                        }

                      } on FirebaseAuthException catch (e) {

                        if (e.code == 'user-not-found' ||

                            e.code == 'wrong-password') {

                          showDialog(

                            context: context,

                            builder: (BuildContext context) {

                              return AlertDialog(

                                title: Text('Error'),

                                content: Text('Incorrect email or password'),

                                actions: [

                                  TextButton(

                                    child: Text('OK'),

                                    onPressed: () {

                                      Navigator.pop(context);

                                    },

                                  ),

                                ],

                              );

                            },

                          );

                        } else {

                          showDialog(

                            context: context,

                            builder: (BuildContext context) {

                              return AlertDialog(

                                title: Text('Error'),

                                content: Text(

                                    'An error occurred while signing in. Please try again later.'),

                                actions: [

                                  TextButton(

                                    child: Text('OK'),

                                    onPressed: () {

                                      Navigator.pop(context);

                                    },

                                  ),

                                ],

                              );

                            },

                          );

                        }

                      }

                    } else {

                      // Invalid email or password

                      showDialog(

                        context: context,

                        builder: (BuildContext context) {

                          return AlertDialog(

                            title: Text('Invalid Email or Password'),

                            content: Text(

                                'Please enter a valid email and password.'),

                            actions: [

                              TextButton(

                                child: Text('OK'),

                                onPressed: () {

                                  Navigator.pop(context);

                                },

                              ),

                            ],

                          );

                        },

                      );

                    }

                  }),

                  signUpOption(),

                ],

              ),

            ),

          ),

        ),

      ),

    );

  }

  bool \_validatePassword(String value) {

    RegExp pattern = RegExp(

      r'^(?=.\*?[A-Z])(?=.\*?[0-9])(?=.\*?[!@#\$&\*~]).{8,}$',

    );

    return pattern.hasMatch(value);

  }

  Row signUpOption() {

    return Row(

      mainAxisAlignment: MainAxisAlignment.center,

      children: [

        const Text(

          "Don't have an account?",

          style: TextStyle(color: Colors.white70),

        ),

        GestureDetector(

          onTap: () {

            Navigator.push(

              context,

              MaterialPageRoute(builder: (context) => SignUpScreen()),

            );

          },

          child: const Text(

            "SIGN UP",

            style: TextStyle(

              color: Colors.white,

              fontWeight: FontWeight.bold,

            ),

          ),

        ),

      ],

    );

  }

  Widget forgetPassword(BuildContext context) {

    return Container(

      width: MediaQuery.of(context).size.width,

      height: 35,

      alignment: Alignment.bottomRight,

      child: TextButton(

        child: const Text(

          "Forgot Password?",

          style: TextStyle(color: Colors.white70),

          textAlign: TextAlign.right,

        ),

        onPressed: () => Navigator.push(

          context,

          MaterialPageRoute(builder: (context) => resetPassword()),

        ),

      ),

    );

  }

}

**SIGNUP\_SCREEN.DART**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:frebase\_signin/reusable\_widgets/reusable\_widget.dart';

import 'package:frebase\_signin/screens/signin\_screen.dart';

class SignUpScreen extends StatefulWidget {

  const SignUpScreen({Key? key}) : super(key: key);

  @override

  State<SignUpScreen> createState() => \_SignUpScreenState();

}

class \_SignUpScreenState extends State<SignUpScreen> {

  TextEditingController \_passwordTextController = TextEditingController();

  TextEditingController \_emailTextController = TextEditingController();

  TextEditingController \_userNameTextController = TextEditingController();

  String? errorMessage;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      extendBodyBehindAppBar: true,

      appBar: AppBar(

        backgroundColor: Colors.transparent,

        elevation: 0,

        title: const Text(

          "SIGN UP",

          style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),

        ),

      ),

      body: Container(

        width: MediaQuery.of(context).size.width,

        height: MediaQuery.of(context).size.height,

        decoration: BoxDecoration(

          gradient: LinearGradient(

            colors: [

              Color.fromARGB(255, 244, 192, 140), // Dark brown

              Color.fromARGB(255, 242, 189, 136), // Medium brown

              Color(0xFFCDBA96), // Light brown

            ],

            begin: Alignment.topCenter,

            end: Alignment.bottomCenter,

          ),

        ),

        child: SingleChildScrollView(

          child: Padding(

            padding: EdgeInsets.fromLTRB(

              20,

              MediaQuery.of(context).size.height \* 0.2,

              20,

              0,

            ),

            child: Column(

              children: <Widget>[

                const SizedBox(

                  height: 20,

                ),

                reusableTextField(

                  "Enter Username",

                  Icons.person\_outline,

                  false,

                  \_userNameTextController,

                ),

                const SizedBox(

                  height: 20,

                ),

                reusableTextField(

                  "Enter Email Id",

                  Icons.person\_outline,

                  false,

                  \_emailTextController,

                ),

                const SizedBox(

                  height: 20,

                ),

                reusableTextField(

                  "Enter Password",

                  Icons.lock,

                  true,

                  \_passwordTextController,

                ),

                Text(

                  'Password must be 8 characters ,contain at least one uppercase letter, one numeric digit, and one special character.',

                  style: TextStyle(color: Colors.grey),

                ),

                const SizedBox(

                  height: 20,

                ),

                firebaseButton(context, "SIGN UP", () async {

                  String password = \_passwordTextController.text;

                  bool isValidPassword = \_validatePassword(password);

                  if (!isValidPassword) {

                    \_showDialog(

                      context,

                      'Invalid Password',

                      'Password must contain at least one uppercase letter, one numeric digit, and one special character.',

                    );

                    return;

                  }

                  try {

                    UserCredential userCredential = await FirebaseAuth.instance

                        .createUserWithEmailAndPassword(

                      email: \_emailTextController.text,

                      password: password,

                    );

                    User? user = userCredential.user;

                    if (user != null) {

                      await user

                          .updateDisplayName(\_userNameTextController.text);

                      print("Created New Account");

                      // Show account created dialog

                      \_showDialog(

                        context,

                        'Account Created',

                        'Your account has been successfully created!',

                      );

                      // Delay navigation to another screen

                      await Future.delayed(const Duration(seconds: 3));

                      Navigator.push(

                        context,

                        MaterialPageRoute(

                          builder: (context) => SignInScreen(),

                        ),

                      );

                    }

                  } catch (error) {

                    String errorMessage = error.toString();

                    if (errorMessage

                        .contains('[firebase\_auth/email-already-in-use] ')) {

                      \_showDialog(

                        context,

                        'Email already exists',

                        'Please use a different email address.',

                      );

                    } else {

                      print("Error: $errorMessage");

                    }

                  }

                }),

                if (errorMessage != null)

                  Text(

                    errorMessage!,

                    style: TextStyle(color: Colors.red),

                  ),

              ],

            ),

          ),

        ),

      ),

    );

  }

  void \_showDialog(BuildContext context, String title, String message) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: Text(title),

          content: Text(message),

          actions: <Widget>[

            TextButton(

              child: Text('OK'),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

          ],

        );

      },

    );

  }

  bool \_validatePassword(String password) {

    // Regex pattern to validate password

    RegExp pattern = RegExp(

      r'^(?=.\*?[A-Z])(?=.\*?[0-9])(?=.\*?[!@#\$&\*~]).{8,}$',

    );

    return pattern.hasMatch(password);

  }

}

**RESET\_PASSWORD.DART**

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:frebase\_signin/reusable\_widgets/reusable\_widget.dart';

class resetPassword extends StatefulWidget {

  const resetPassword({super.key});

  @override

  State<resetPassword> createState() => \_resetPasswordState();

}

class \_resetPasswordState extends State<resetPassword> {

  TextEditingController \_emailTextController = TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      extendBodyBehindAppBar: true,

      appBar: AppBar(

        backgroundColor: Colors.transparent,

        elevation: 0,

        title: const Text(

          "Reset Password",

          style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),

        ),

      ),

      body: Container(

          width: MediaQuery.of(context).size.width,

          height: MediaQuery.of(context).size.height,

          decoration: BoxDecoration(

              gradient: LinearGradient(colors: [

            Color.fromARGB(255, 244, 192, 140), // Dark brown

            Color.fromARGB(255, 242, 189, 136), // Medium brown

            Color(0xFFCDBA96), // Light brown

          ], begin: Alignment.topCenter, end: Alignment.bottomCenter)),

          child: SingleChildScrollView(

            child: Padding(

                padding: EdgeInsets.fromLTRB(

                    20, MediaQuery.of(context).size.height \* 0.2, 20, 0),

                child: Column(

                  children: <Widget>[

                    const SizedBox(

                      height: 20,

                    ),

                    reusableTextField("Enter Email Id", Icons.person\_outline,

                        false, \_emailTextController),

                    const SizedBox(

                      height: 20,

                    ),

                    firebaseButton(context, "Reset Password", () {

                      FirebaseAuth.instance

                          .sendPasswordResetEmail(

                              email: \_emailTextController.text)

                          .then((value) => Navigator.of(context).pop());

                    })

                  ],

                )),

          )),

    );

  }

}

**HOME\_SCREEN.DART**

import 'package:carousel\_slider/carousel\_slider.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'DetailScreen\_page.dart';

class HomeScreen extends StatefulWidget {

  const HomeScreen({Key? key, required this.username}) : super(key: key);

  final String username;

  @override

  \_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

  List<Map<String, dynamic>> stylistData = [

    {

      'stylistName': 'Jones',

      'salonName': 'Sessor Sounds',

      'rating': '4.8',

      'rateAmount': '56',

      'imgUrl': 'assets/images/stylist1.png',

      'bgColor': Color(0xffFFF0EB),

    },

    {

      'stylistName': 'Robert',

      'salonName': ' Sessor Sounds',

      'rating': '4.7',

      'rateAmount': '80',

      'imgUrl': 'assets/images/stylist2.png',

      'bgColor': Color(0xffEBF6FF),

    },

    {

      'stylistName': 'Stokes',

      'salonName': 'Sessors Sound',

      'rating': '4.7',

      'rateAmount': '70',

      'imgUrl': 'assets/images/stylist3.png',

      'bgColor': Color(0xffFFF3EB),

    }

  ];

  @override

  Widget build(BuildContext context) {

    return WillPopScope(

      onWillPop: () async {

        SystemNavigator.pop();

        return false;

      },

      child: Scaffold(

        backgroundColor: Color.fromARGB(255, 251, 251, 251),

        appBar: AppBar(

          title: Text('Home'),

          actions: [

            IconButton(

              icon: Icon(Icons.logout),

              onPressed: () {

                FirebaseAuth.instance.signOut().then((value) {

                  print("Signed out");

                  Navigator.pushNamedAndRemoveUntil(

                    context,

                    '/signin',

                    (route) => false,

                  );

                });

              },

            ),

          ],

        ),

        body: SingleChildScrollView(

          child: SafeArea(

            child: Column(

              children: [

                SizedBox(height: 30),

                Padding(

                  padding: EdgeInsets.symmetric(horizontal: 20),

                  child: Text(

                    'LIVE OFFERS ',

                    style: TextStyle(

                      fontWeight: FontWeight.bold,

                      fontSize: 24,

                      color: Color.fromARGB(255, 255, 156, 156),

                    ),

                  ),

                ),

                SizedBox(height: 10),

                Container(

                  decoration: BoxDecoration(

                    border: Border.all(

                      color: const Color.fromARGB(255, 232, 174, 174),

                      width: 0,

                    ),

                  ),

                  child: CarouselSlider(

                    options: CarouselOptions(

                      height: 200,

                      autoPlay: true,

                      aspectRatio: BorderSide.strokeAlignCenter,

                      enlargeCenterPage: true,

                      viewportFraction: 0.3,

                    ),

                    items: [

                      Image.asset('assets/images/Banner.png', fit: BoxFit.fill),

                      Image.asset('assets/images/Banner1.jpeg',

                          fit: BoxFit.cover),

                      Image.asset('assets/images/Banner2.jpeg',

                          fit: BoxFit.cover),

                      Image.asset('assets/images/Banner3.jpg',

                          fit: BoxFit.cover),

                      Image.asset('assets/images/Banner4.jpg',

                          fit: BoxFit.cover),

                    ],

                  ),

                ),

                SizedBox(height: 30),

                Container(

                  height: MediaQuery.of(context).size.height,

                  width: MediaQuery.of(context).size.width,

                  decoration: BoxDecoration(

                    color: Colors.white,

                    borderRadius: BorderRadius.vertical(

                      top: Radius.circular(50),

                    ),

                  ),

                  child: Padding(

                    padding: EdgeInsets.symmetric(horizontal: 30),

                    child: Column(

                      crossAxisAlignment: CrossAxisAlignment.start,

                      children: [

                        SizedBox(height: 50),

                        Text(

                          'Hair Stylist',

                          style: TextStyle(

                            fontWeight: FontWeight.bold,

                            fontSize: 24,

                          ),

                        ),

                        Column(

                          children: stylistData.map((stylist) {

                            return StylistCard(stylist);

                          }).toList(),

                        ),

                      ],

                    ),

                  ),

                ),

              ],

            ),

          ),

        ),

      ),

    );

  }

}

class StylistCard extends StatelessWidget {

  final Map<String, dynamic> stylist;

  StylistCard(this.stylist);

  @override

  Widget build(BuildContext context) {

    return Container(

      margin: EdgeInsets.symmetric(vertical: 20),

      width: MediaQuery.of(context).size.width,

      height: MediaQuery.of(context).size.height / 4 - 20,

      decoration: BoxDecoration(

        borderRadius: BorderRadius.circular(20),

        color: stylist['bgColor'],

      ),

      child: Stack(

        children: <Widget>[

          Positioned(

            top: 20,

            right: -60,

            child: Image.asset(

              stylist['imgUrl'],

              width: MediaQuery.of(context).size.width \* 0.60,

            ),

          ),

          Padding(

            padding: EdgeInsets.only(top: 40, left: 30),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              children: <Widget>[

                Text(

                  stylist['stylistName'],

                  style: TextStyle(

                    fontWeight: FontWeight.w600,

                    fontSize: 20,

                  ),

                ),

                SizedBox(

                  height: 5,

                ),

                Text(

                  stylist['salonName'],

                  style: TextStyle(

                    fontWeight: FontWeight.w300,

                  ),

                ),

                SizedBox(

                  height: 10,

                ),

                Row(

                  children: <Widget>[

                    Icon(

                      Icons.star,

                      size: 16,

                      color: Color(0xff4E295B),

                    ),

                    SizedBox(

                      width: 10,

                    ),

                    Text(

                      stylist['rating'],

                      style: TextStyle(

                        color: Color(0xff4E295B),

                      ),

                    ),

                  ],

                ),

                SizedBox(

                  height: 20,

                ),

                MaterialButton(

                  onPressed: () {

                    Navigator.push(

                      context,

                      MaterialPageRoute(

                        builder: (context) => DetailScreen(stylist),

                      ),

                    );

                  },

                  color: Color.fromARGB(255, 91, 56, 41),

                  shape: RoundedRectangleBorder(

                    borderRadius: BorderRadius.circular(20),

                  ),

                  child: Text(

                    'View Profile',

                    style: TextStyle(

                      color: Colors.white,

                    ),

                  ),

                ),

              ],

            ),

          ),

        ],

      ),

    );

  }

}

**DEATIL\_SCREEN.DART**

import 'package:flutter/material.dart';

import 'package:frebase\_signin/screens/BookingDeatils.dart';

void main() {

  runApp(MaterialApp(

    home: DetailScreen({

      'stylistName': 'Cameron Thomas',

      'salonName': 'Sesscors Sounds',

      'rating': '4.5',

      'rateAmount': '125',

      'bgColor': Color(0xffF7F5F9),

      'imgUrl': 'assets/images/stylist1.png',

    }),

  ));

}

var serviceList = [

  {'title': 'Men\'s Hair Cut', 'duration': 45, 'price': 100},

  {'title': 'Shaving', 'duration': 25, 'price': 80},

  {'title': 'Color & Blow Dry', 'duration': 90, 'price': 300},

  {'title': 'Oil Treatment', 'duration': 30, 'price': 70},

];

class DetailScreen extends StatelessWidget {

  final stylist;

  DetailScreen(this.stylist);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: SingleChildScrollView(

        child: Container(

          width: MediaQuery.of(context).size.width,

          height: MediaQuery.of(context).size.height,

          child: Stack(

            children: <Widget>[

              Container(

                height: MediaQuery.of(context).size.height / 3 + 20,

                width: MediaQuery.of(context).size.width,

                child: Stack(

                  fit: StackFit.expand,

                  children: <Widget>[

                    Image.asset(

                      'assets/images/detail\_bg.png',

                      fit: BoxFit.fill,

                    ),

                    Container(

                      width: MediaQuery.of(context).size.width,

                      height: MediaQuery.of(context).size.height,

                      color: Colors.purple.withOpacity(0.1),

                    ),

                  ],

                ),

              ),

              Positioned(

                top: 50,

                left: 20,

                child: IconButton(

                  icon: Icon(

                    Icons.arrow\_back\_ios,

                    color: Colors.white,

                  ),

                  onPressed: () {

                    Navigator.pop(context);

                  },

                ),

              ),

              Positioned(

                top: MediaQuery.of(context).size.height / 3 - 30,

                child: Container(

                  width: MediaQuery.of(context).size.width,

                  height: MediaQuery.of(context).size.height,

                  decoration: BoxDecoration(

                    color: Colors.white,

                    borderRadius: BorderRadius.only(

                      topLeft: Radius.circular(50),

                    ),

                  ),

                  child: Padding(

                    padding: EdgeInsets.symmetric(horizontal: 30),

                    child: Column(

                      crossAxisAlignment: CrossAxisAlignment.start,

                      children: <Widget>[

                        SizedBox(

                          height: 80,

                        ),

                        Text(

                          'Service List',

                          style: TextStyle(

                            fontWeight: FontWeight.bold,

                            fontSize: 25,

                          ),

                        ),

                        SizedBox(height: 20),

                        ServiceTile(

                          serviceList[0],

                          Color.fromARGB(255, 225, 219, 219),

                        ),

                        ServiceTile(

                          serviceList[1],

                          Color.fromARGB(255, 225, 219, 219),

                        ),

                        ServiceTile(

                          serviceList[2],

                          Color.fromARGB(255, 225, 219, 219),

                        ),

                        ServiceTile(

                          serviceList[3],

                          Color.fromARGB(255, 225, 219, 219),

                        ),

                      ],

                    ),

                  ),

                ),

              ),

              Positioned(

                top: MediaQuery.of(context).size.height / 3 - 120,

                child: Padding(

                  padding: EdgeInsets.symmetric(horizontal: 30),

                  child: Row(

                    crossAxisAlignment: CrossAxisAlignment.end,

                    children: <Widget>[

                      Container(

                        width: MediaQuery.of(context).size.width / 3 - 20,

                        height: MediaQuery.of(context).size.height / 6 + 20,

                        decoration: BoxDecoration(

                          color: stylist['bgColor'],

                          borderRadius: BorderRadius.circular(20),

                        ),

                        child: Stack(

                          fit: StackFit.expand,

                          children: <Widget>[

                            Positioned(

                              top: 10,

                              right: -25,

                              child: Image.asset(

                                stylist['imgUrl'],

                                scale: 1.7,

                              ),

                            ),

                          ],

                        ),

                      ),

                      SizedBox(

                        width: 20,

                      ),

                      Column(

                        crossAxisAlignment: CrossAxisAlignment.start,

                        children: <Widget>[

                          Text(

                            stylist['stylistName'],

                            style: TextStyle(

                              fontWeight: FontWeight.bold,

                              fontSize: 20,

                            ),

                          ),

                          SizedBox(

                            height: 0,

                          ),

                          Text(

                            stylist['salonName'],

                            style: TextStyle(

                              fontWeight: FontWeight.w300,

                              color: Colors.grey,

                            ),

                          ),

                          SizedBox(

                            height: 10,

                          ),

                          Row(

                            children: <Widget>[

                              Icon(

                                Icons.star,

                                size: 16,

                                color: Color(0xffFF8573),

                              ),

                              SizedBox(width: 5),

                              Text(

                                stylist['rating'],

                                style: TextStyle(

                                  color: Color(0xffFF8573),

                                ),

                              ),

                              SizedBox(

                                width: 5,

                              ),

                              Text(

                                '(${stylist['rateAmount']})',

                                style: TextStyle(

                                  color: Colors.grey,

                                ),

                              ),

                            ],

                          )

                        ],

                      ),

                    ],

                  ),

                ),

              ),

            ],

          ),

        ),

      ),

    );

  }

}

class ServiceTile extends StatelessWidget {

  final service;

  final Color backgroundColor;

  ServiceTile(this.service, this.backgroundColor);

  @override

  Widget build(BuildContext context) {

    return Container(

      margin: EdgeInsets.only(bottom: 30),

      padding: EdgeInsets.all(20),

      decoration: BoxDecoration(

        color: backgroundColor,

        borderRadius: BorderRadius.circular(10),

      ),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceBetween,

        children: <Widget>[

          Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: <Widget>[

              Text(

                service['title'],

                style: TextStyle(

                  fontSize: 16,

                  fontWeight: FontWeight.w500,

                ),

              ),

              SizedBox(

                height: 5,

              ),

              Text(

                '${service['duration']} mins',

                style: TextStyle(

                  color: const Color.fromARGB(255, 89, 87, 87),

                ),

              ),

            ],

          ),

          Row(

            children: [

              Text(

                '₹${service['price']}',

                style: TextStyle(

                  fontSize: 15,

                  fontWeight: FontWeight.bold,

                ),

              ),

              SizedBox(width: 10),

              ElevatedButton(

                onPressed: () {

                  showDialog(

                    context: context,

                    builder: (context) => BookingDialog(service),

                  );

                },

                style: ElevatedButton.styleFrom(

                  primary: Color.fromARGB(255, 154, 129, 120),

                ),

                child: Text('Book Now'),

              ),

            ],

          ),

        ],

      ),

    );

  }

}

**ORDERSUMMARY.DART**

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'PaymentOptions\_page.dart';

class OrderSummaryPage extends StatelessWidget {

  final service;

  final String name;

  final String phoneNumber;

  final String date;

  final String time;

  final String stylistName;

  const OrderSummaryPage(

    this.service,

    this.name,

    this.phoneNumber,

    this.date,

    this.time,

    this.stylistName,

  );

  void \_storeOrderSummaryData() async {

    FirebaseFirestore firestore = FirebaseFirestore.instance;

    DocumentReference serviceRef =

        await firestore.collection('Service Details').add({

      'title': service['title'],

      'duration': service['duration'],

      'price': service['price'],

    });

    print('service details stored successfully');

    DocumentReference customerRef =

        await firestore.collection('Customer Details').add({

      'serviceId': serviceRef.id,

      'name': name,

      'phoneNumber': phoneNumber,

    });

    print('customer details stored successfully');

    await firestore.collection('Stylist Name').add({

      'serviceId': serviceRef.id,

      'stylistName': stylistName,

    });

    print('stylist name stored successfully');

    await firestore.collection('Booking Details').add({

      'serviceId': serviceRef.id,

      'customerId': customerRef.id,

      'date': date,

      'time': time,

    });

    print('booking details stored successfully');

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Order Summary'),

      ),

      body: Padding(

        padding: EdgeInsets.all(20.0),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Text(

              'Service Details',

              style: TextStyle(

                fontWeight: FontWeight.bold,

                fontSize: 20.0,

              ),

            ),

            SizedBox(height: 10),

            Text('Title: ${service['title']}'),

            Text('Duration: ${service['duration']} mins'),

            Text('Price: ₹${service['price']}'),

            SizedBox(height: 20),

            Text(

              'Customer Details',

              style: TextStyle(

                fontWeight: FontWeight.bold,

                fontSize: 20.0,

              ),

            ),

            SizedBox(height: 10),

            Text('Name: $name'),

            Text('Phone Number: $phoneNumber'),

            const SizedBox(height: 20),

            const Text(

              'Stylist Name',

              style: TextStyle(

                fontWeight: FontWeight.bold,

                fontSize: 20.0,

              ),

            ),

            const SizedBox(height: 10),

            Text('Stylist Name: $stylistName'),

            const SizedBox(height: 20),

            const Text(

              'Booking Details',

              style: TextStyle(

                fontWeight: FontWeight.bold,

                fontSize: 20.0,

              ),

            ),

            const SizedBox(height: 10),

            Text('Date: $date'),

            Text('Time: $time'),

            const SizedBox(height: 20),

            Expanded(

              child: Align(

                alignment: Alignment.bottomCenter,

                child: ElevatedButton(

                  onPressed: () {

                    \_storeOrderSummaryData();

                    Navigator.push(

                      context,

                      MaterialPageRoute(builder: (context) => PaymentPage()),

                    );

                    // Add your payment logic here

                  },

                  style: ElevatedButton.styleFrom(

                    backgroundColor: Colors.brown,

                    padding: EdgeInsets.symmetric(vertical: 20, horizontal: 40),

                    shape: RoundedRectangleBorder(

                      borderRadius: BorderRadius.circular(30.0),

                    ),

                  ),

                  child: const Text('Make Payment'),

                ),

              ),

            ),

          ],

        ),

      ),

    );

  }

}

**PAYMENT\_OPTIONS\_PAGE.DART**

import 'package:flutter/material.dart';

import 'package:frebase\_signin/screens/Upi\_Payments.dart';

class PaymentPage extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Payment'),

      ),

      body: Padding(

        padding: EdgeInsets.all(16.0),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Text(

              'Payment Options',

              style: TextStyle(

                fontWeight: FontWeight.bold,

                fontSize: 24.0,

              ),

            ),

            Divider(),

            ListTile(

              leading: Icon(Icons.payment),

              title: Text('UPI'),

              onTap: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                    builder: (context) => const Payment\_Page(

                      name: '',

                      phoneNumber: '',

                      amount: 200.00,

                    ),

                  ),

                );

                // Handle UPI payment option

              },

            ),

            Divider(),

            ListTile(

              leading: Icon(Icons.monetization\_on),

              title: Text('Pay on Spot'),

              onTap: () {

                Navigator.push(

                  context,

                  MaterialPageRoute(

                    builder: (context) => const Payment\_Page(

                      name: '',

                      phoneNumber: '',

                      amount: 200.00,

                    ),

                  ),

                );

                // Handle cash on delivery payment option

              },

            ),

          ],

        ),

      ),

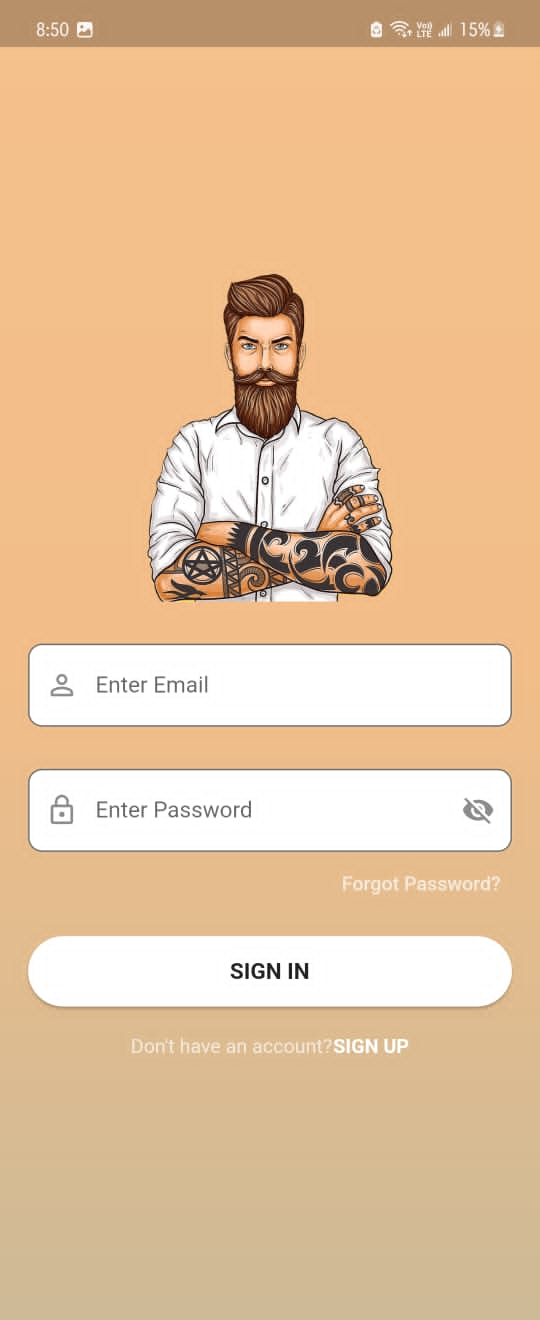
    );

  }

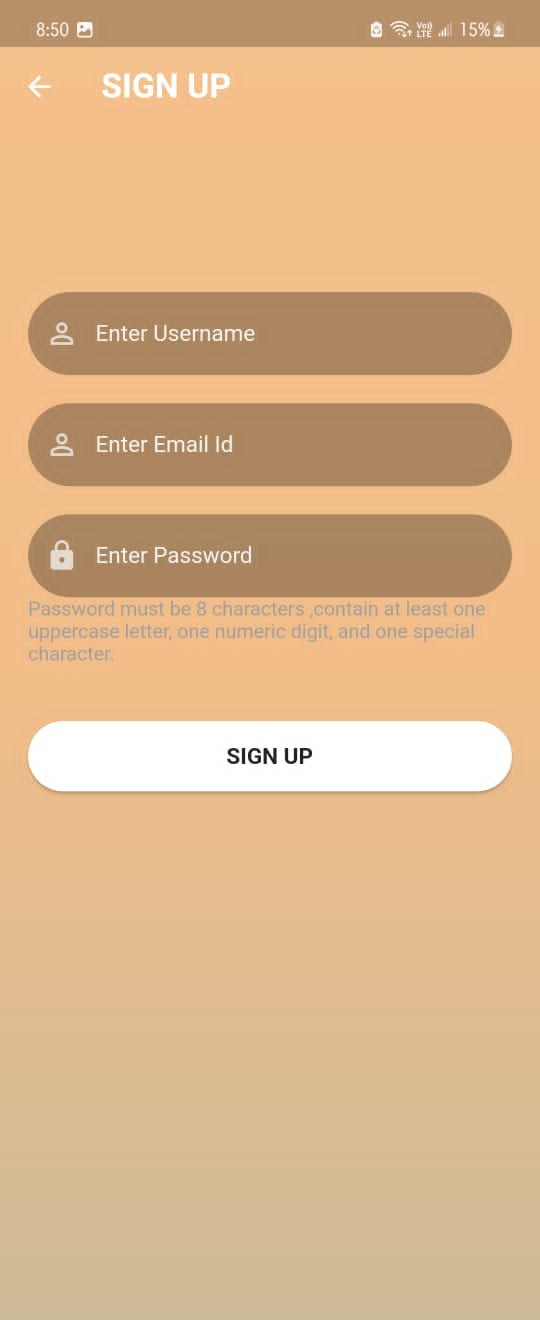
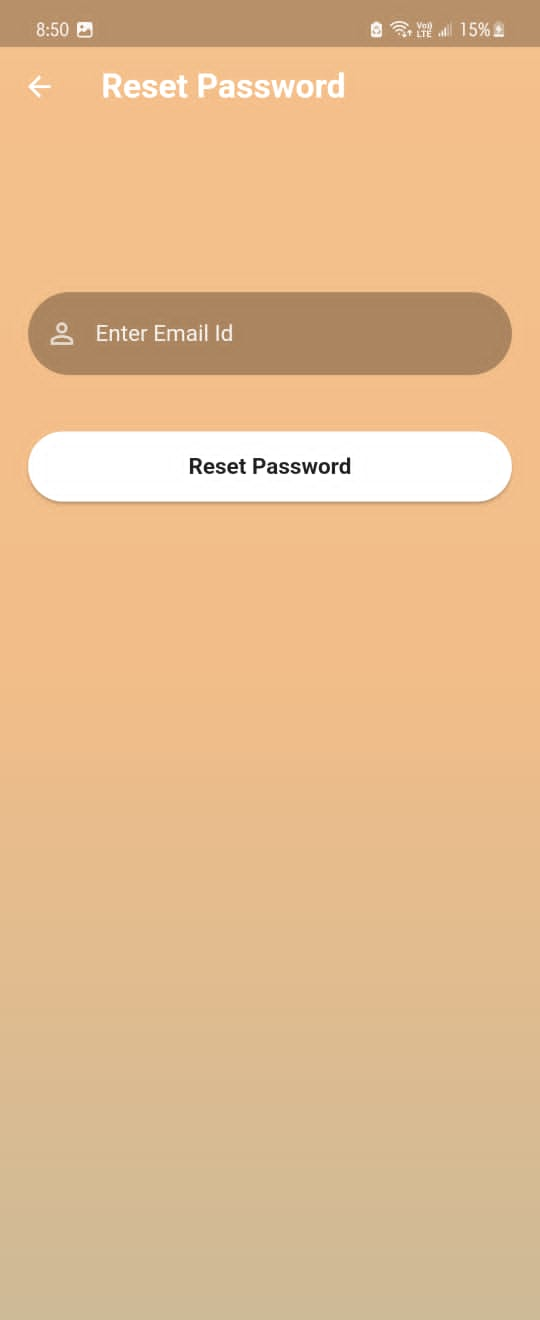
}

**9.USER INTERFACE**

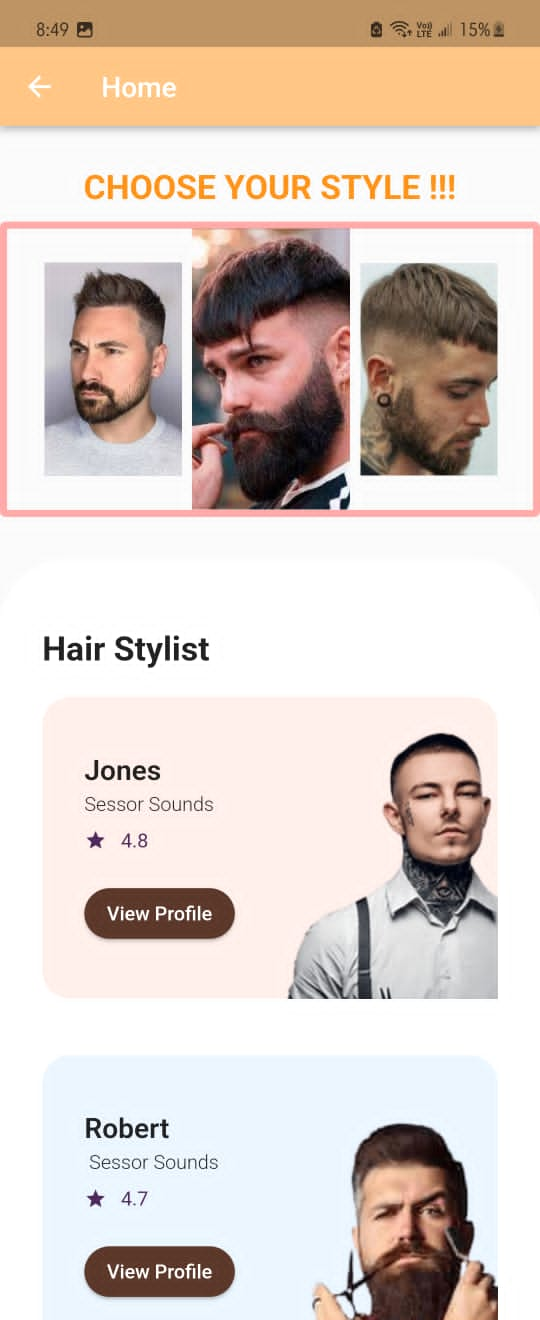
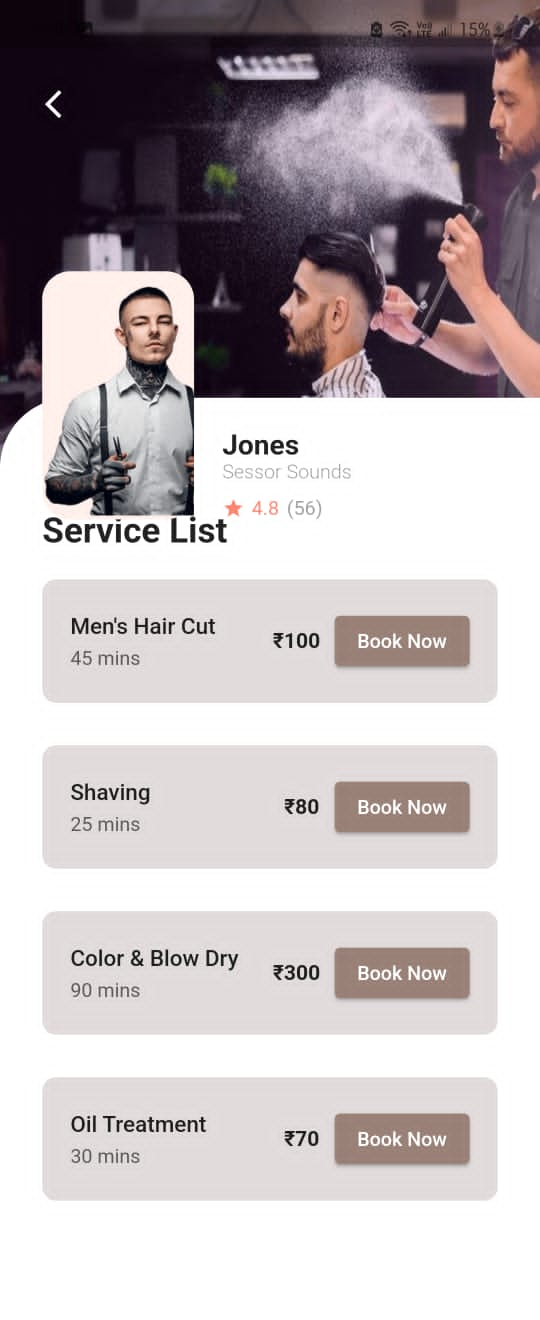
**Splash Screen User Sign In**

** **

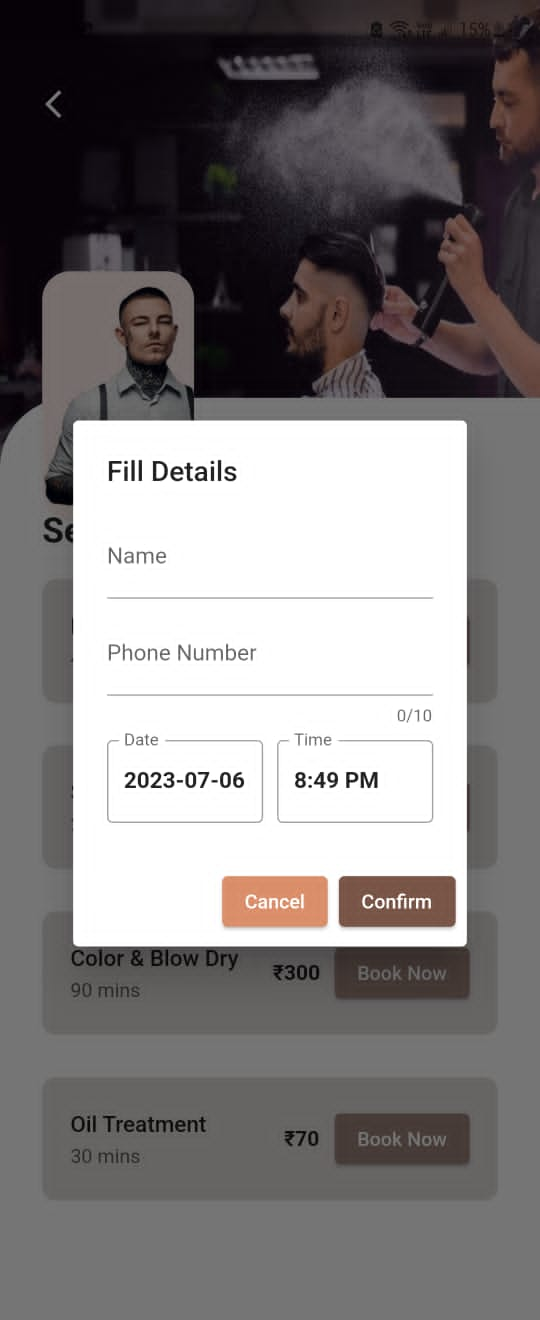
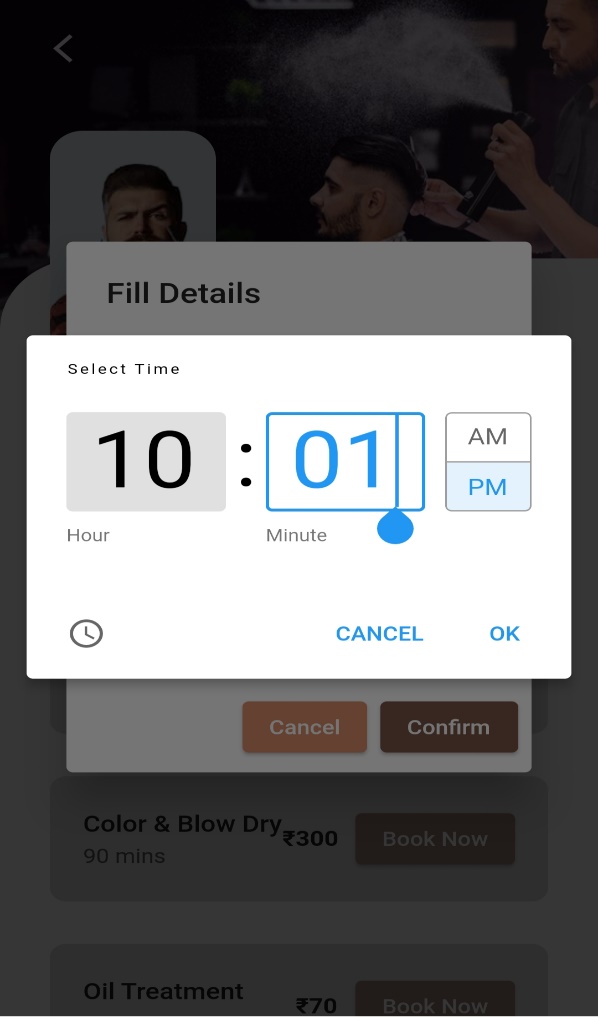
**Sign Up Forgot Password**

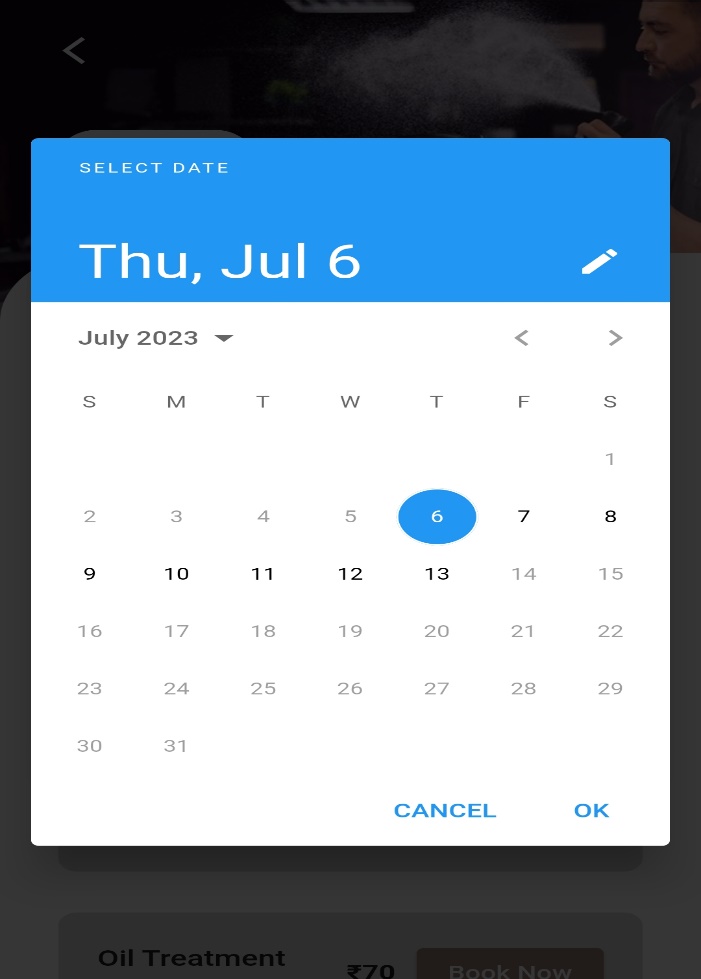
** **

**Home Service List**

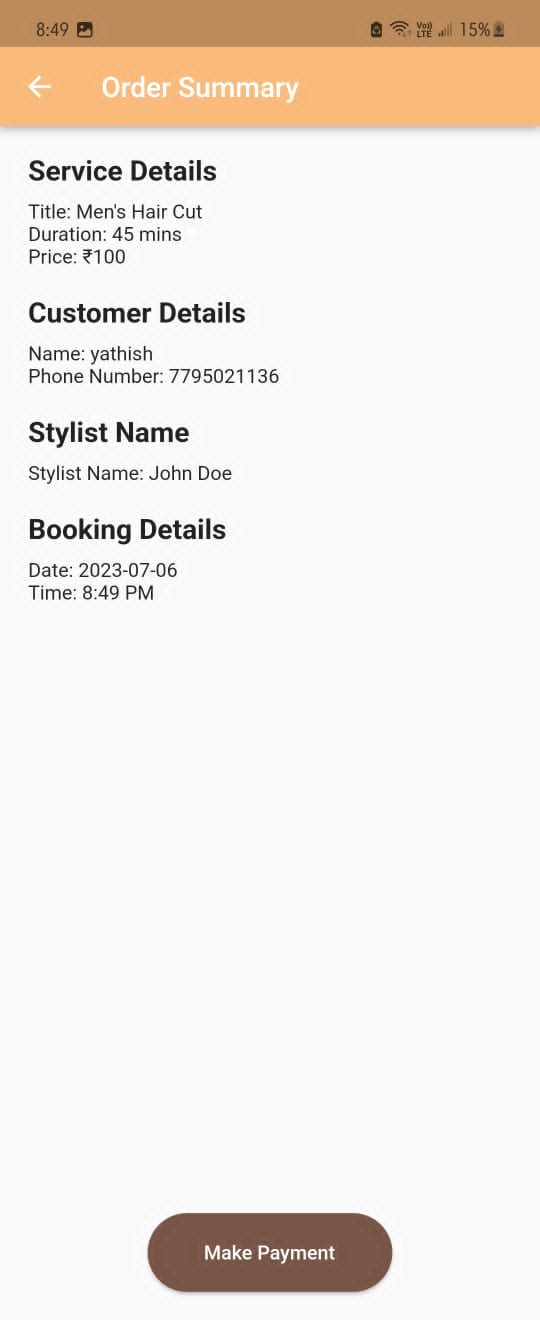
** **

**Service Appointment Pages**

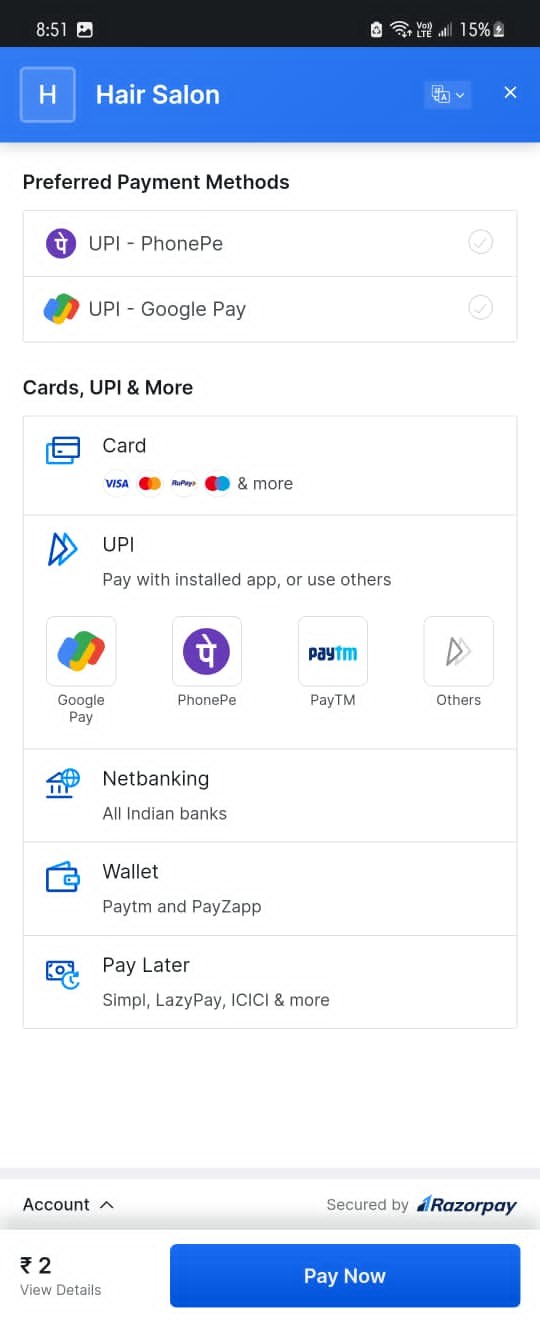
** **

****

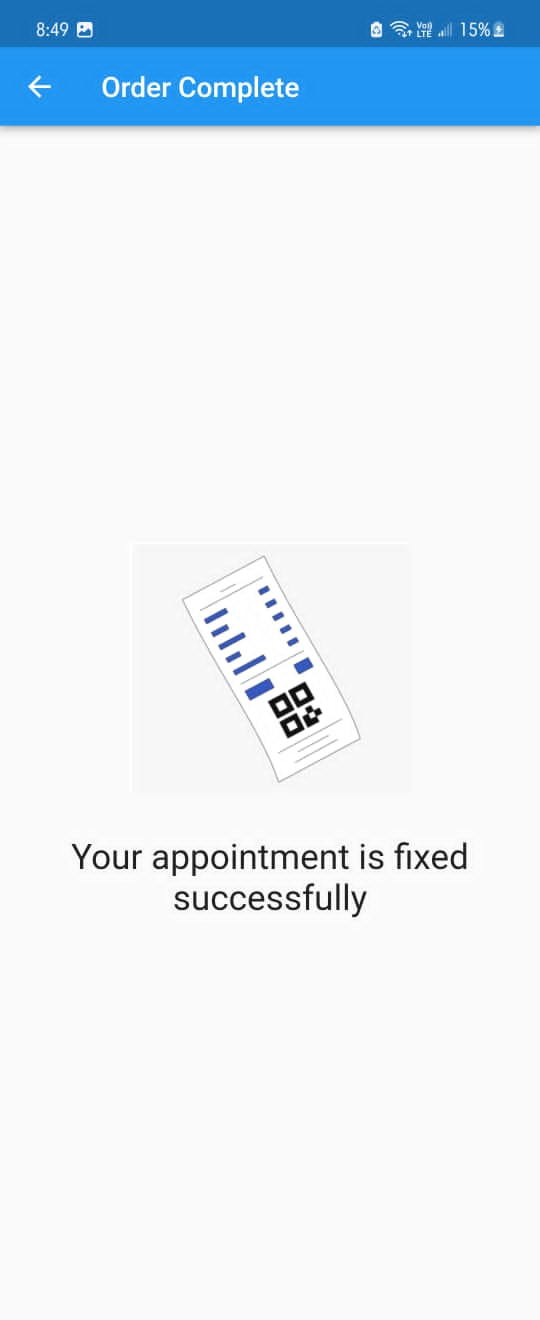
**Order Summary Payments**

** **

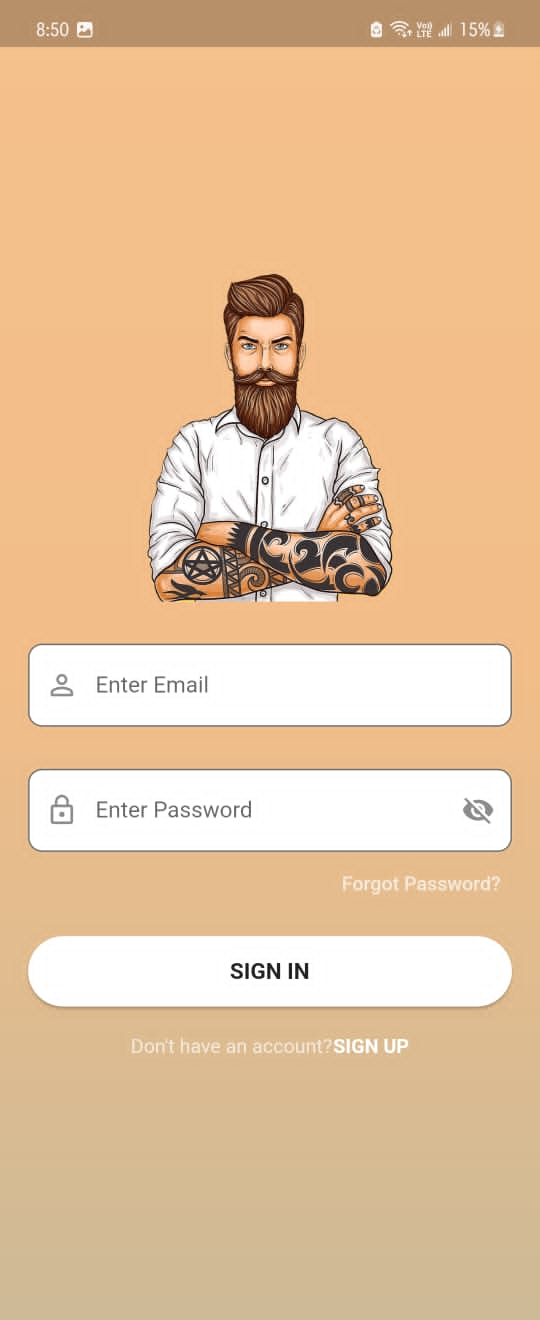
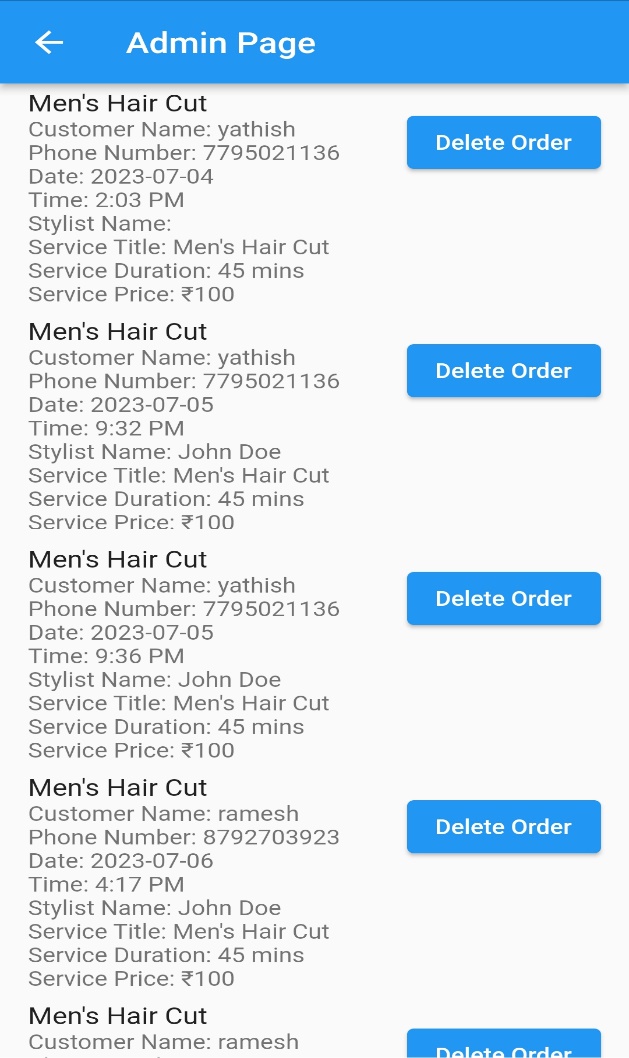
**Online Payment**

** **

**Cash Payment**

****

**Admin Login Delete Order**

** **

**10. TESTING AND IMPLEMENTATION**

Testing and implementation is the process, which tells the reality efficiency and the flexibility of the system design. Reliability means how much the user is expecting from the system. Flexibility tells how much the user is comfortable and hopes additional facilities with the system.

**Testing**

Testing is vital to the success of the system. System testing makes a logical assumption that if all the part of the system is correct, the goal will be successively achieved. It is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing presents interesting anomaly of the software. The testing phase involves testing of system using various test data.

The first test of system to see whether it produces the correct output. When the software is tested the actual output is tested with the expected output. If there is a discrepancy the sequence of instruction must be traced to determine the problem. Breaking the program down into self-contained portions, each of which can be checked at certain key points facilities the process.

The best program is worthless if it does not meet needs. The first step n system testing is to prepare a plan that will test all aspects of the system in a way that promotes its credibility among potential users. The design phase focuses on the detail implementation of the system recommended in the feasibility study. Emphasis is on translating performance specification into design specification. The design phase is a transition from a user-oriented document to document oriented to the programmers or database personnel.

System design goes through two phases of development, logical and physical design. The logical design describes the input, output, database and procedures. Example: Dataflow Diagram. The physical design procures the working system by defining specification that tells the programmers exactly that what the candidates system must do.

The development of software system involves a series of production activities where opportunities for injunction of human error are enormous. Error may occur at the very imperfectly specified as well as later design and development stages. Because of human inability to perform and perfection, software development is followed by a quantity assurance activity.

Quantity assurance also places a vital role in the whole development of the system. The quantity assurance whole of the testing phase is the testing phase is to assure that completeness accuracy of the system and minimize the testing process. In the implementation phase, the goal is to provide a logical order of the testing the common view is to eliminate program errors. This is extremely difficult and time consuming since designer cannot prove 100% accuracy. Therefore, all that can be one is to put system through a “fail test” cycle-determine what will make it fail. A successful, then is one that find errors.

**System Testing**

Testing is an important phase in the development life cycle of the product. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program is performing as expected. Errors were found and corrected by using the various testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system before it was ready for implementation. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

**Testing Levels**

Testing is part of Verification and Validation. Testing plays a very critical role for quality assurance and for ensuring the reliability of the software. The objective of testing can be stated in the following ways.

• A successful test is one that uncovers as-yet-undiscovered bugs.

• A better test case has high probability of finding un-noticed bugs.

• A pessimistic approach of running the software with the intent of finding errors.

• Testing can be performed in various levels like unit test, integration test and system test.

**Unit Testing**

Unit testing tests the individual components to ensure that they operate correctly. Each component is tested independently, without other system component. Each unit was tested with the set of proper test data and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module. There are mainly three different modules in the system. The module 1 which contains most of the user operations was performed by entering and storing related data. Second module consist of staff operations such as adding city, area and managing rates, etc. Testing was done by giving appropriate values. The third module which contains transaction management is done by reducing the amount from accounts manually.

**Integration Testing**

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together and tested in small segment, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole. First the integration between transaction module and staff’s module is integrated and performed a transaction using the interface given by staff’s module. Integration of user’s module is done after this. Which allows to check if the previous integration is working properly along with user module.

**System Testing**

System testing tests a completely integrated system to verify that it meets its requirements. The system testing is started by staff adding cities and areas of the city to database and adding rates according to vehicle type. The user registration is done by giving necessary personal and contact details after that user adds vehicle information and documents to his account for verification by administrator at the office. The admin checks the details and accepts or rejects the vehicle. If it is verified successfully then an NFC tag is allocated to the user. Transactions is done by scanning NFC tag of the user it successfully reduced corresponding amount from user’s account and necessary records kept about the transaction in both user’s and systems database.

**Sub-system Testing**

This phase involves testing collections of modules which have been integrated into subsystems. Sub-systems may be independently designed and implemented. The most common problems which arise in large software systems are sub-system interface mismatches. The subsystem test process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

**Acceptance Testing**

This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system procurer rather than simulated test data. Acceptance test may reveal errors and omissions in the system requirements definition because the real data exercises the system in different ways from the test data. Acceptance testing may also reveal requirement problems where the system facilities do not really meet the user needs or the system performance is unacceptable. The final system is tested for the end user’s data, whether it meets the initial user requirements, whether some of the requirements have been omitted and whether the performance of the system is efficient as a whole.

**Test case 1**

Test Objective: Test for email and Password information

Test Data: Valid: Enter valid email and password. Click on login

Invalid: When incorrect email and password enter.

Output: Valid: If User login to application he can access all services.

If admin sign in he can access the Appointment records.

Invalid: shows the error message.

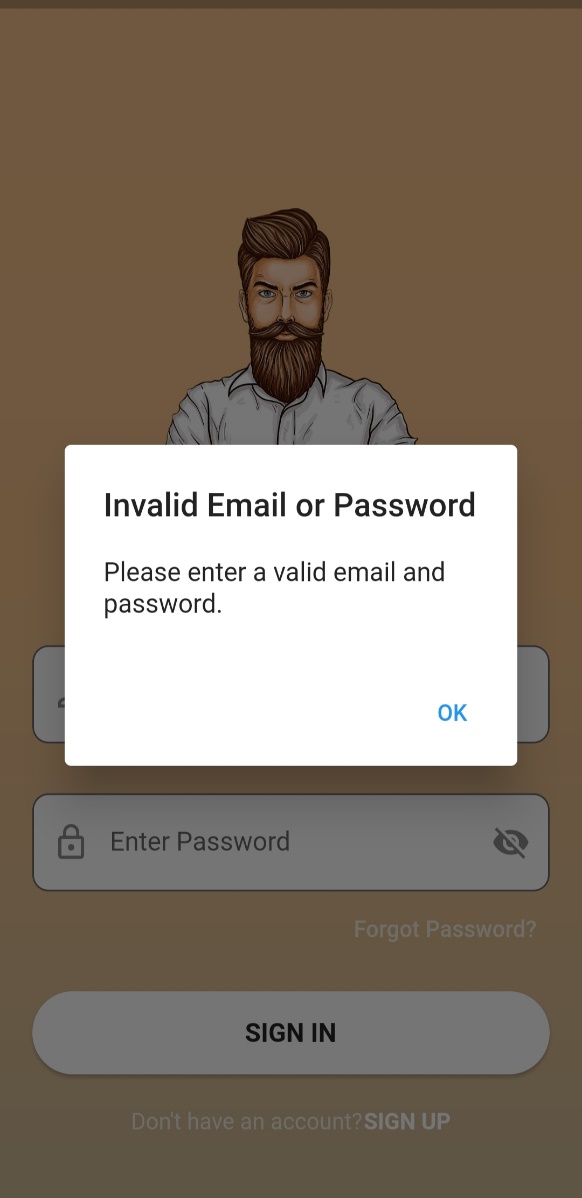
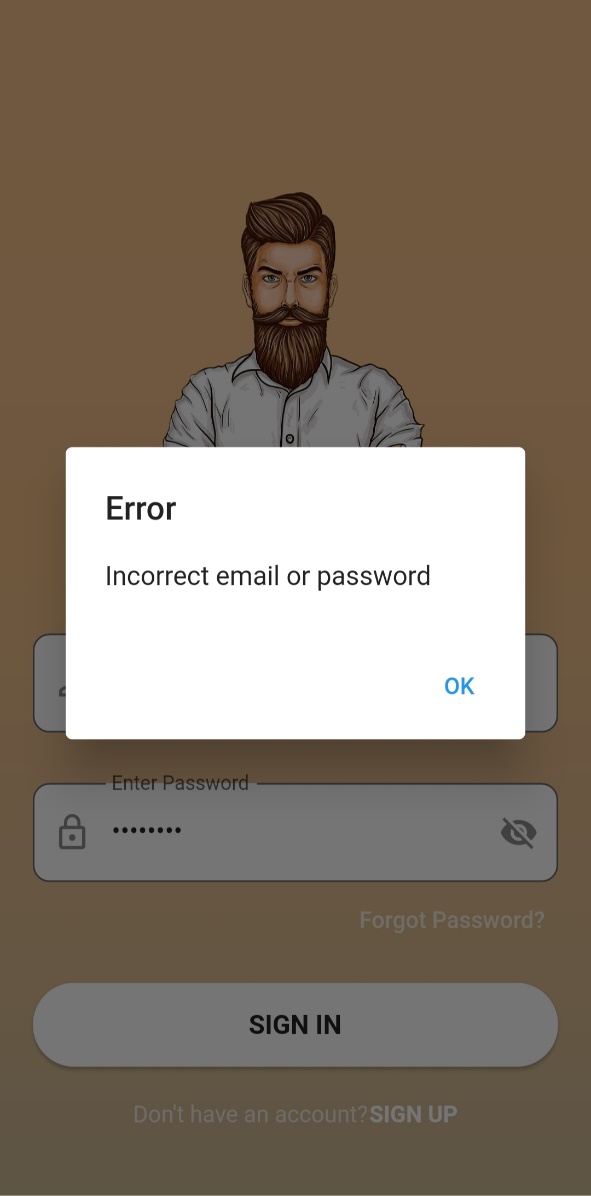
Result: Valid: the user sign in successfully and allowed to enter app.

Invalid: the user prompt with an error message and restricted

to enter the application.

Conclusion: both the valid and invalid result is tested. Output tally

with the required result, hence the test is successfully.

**Test case 2**

Test Objective: Test for Sign Up information.

Test Data: Valid: Enter all the required fields.

Invalid: shows the error message.

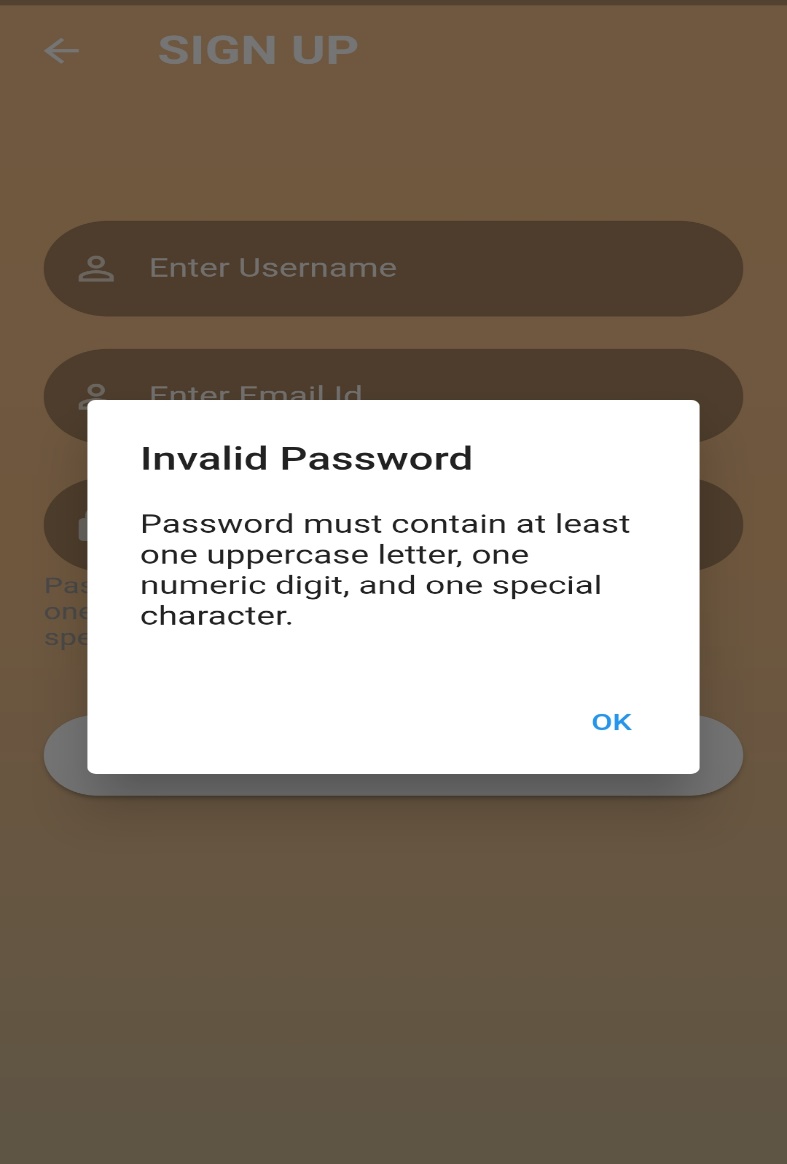
Output: Valid: Allow all records to be added to the database.

Invalid: shows the error message.

Result: Valid: record will be saved

Conclusion: both the valid and invalid result is tested. Output tally

with the required result ,hence the test is successful.



**Test case 3**

Test Objective: Test for book an appointment date .

Test Data: Valid: Select pre-book an appointment of one week from

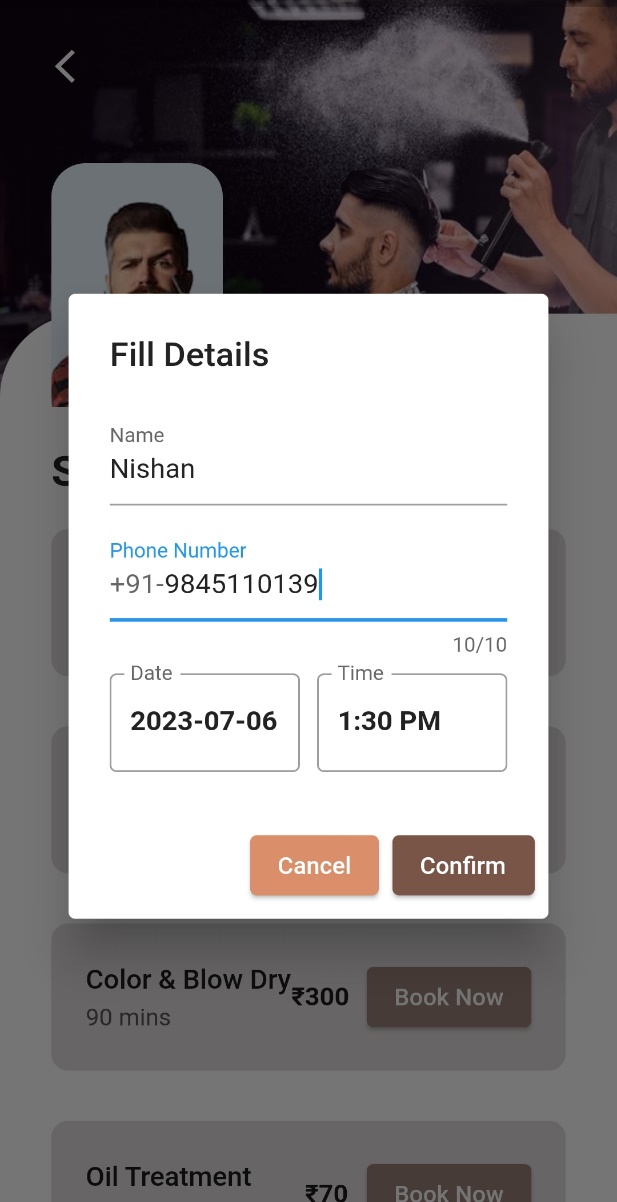
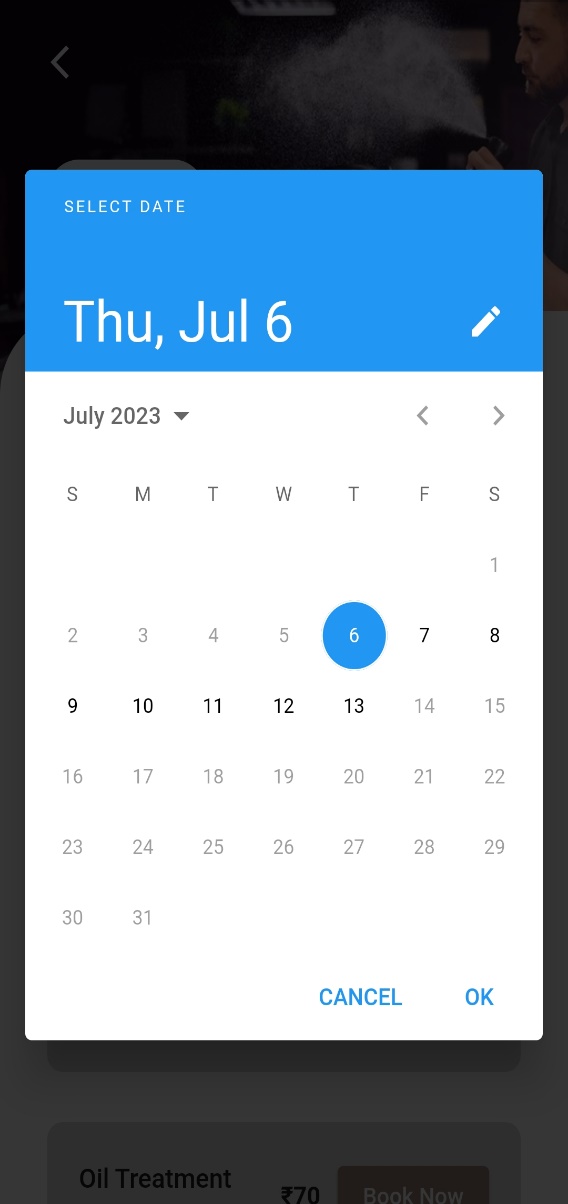
present calendar date.

Invalid: Cannot select after one week or yesterday dates.

Output: Valid: Allow date to upload to database.

Invalid: Fail to upload date to database.

Result : Valid: Date is select to upload for database.

**Test case 4**

Test Objective: Test for book an appointment time.

Test Data: Valid: Select a time between 10 AM and 10 PM.

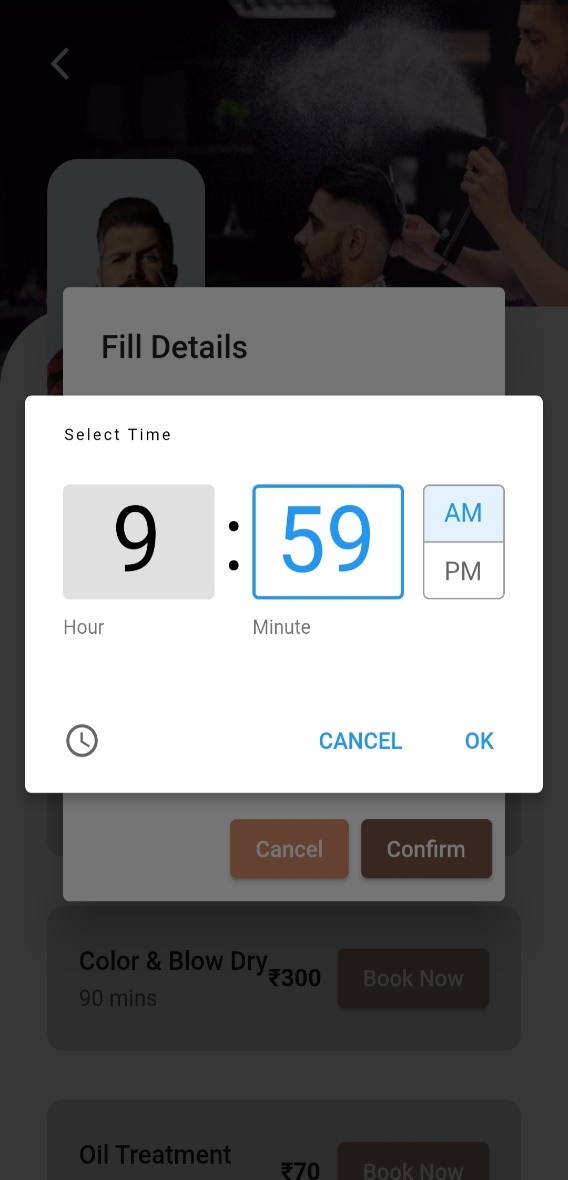
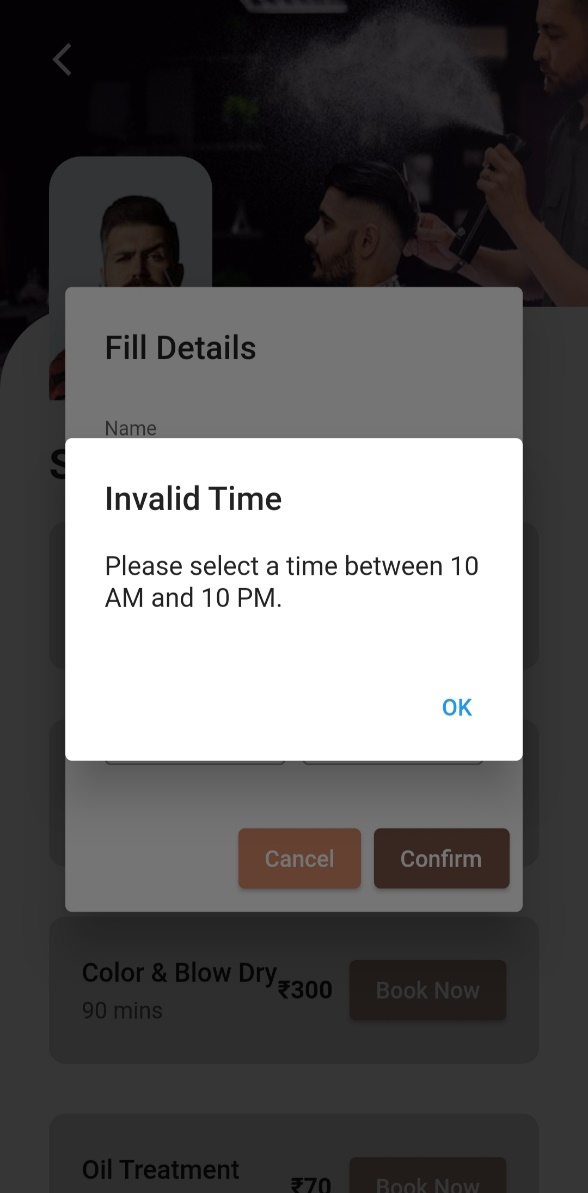
Invalid: Error message shows

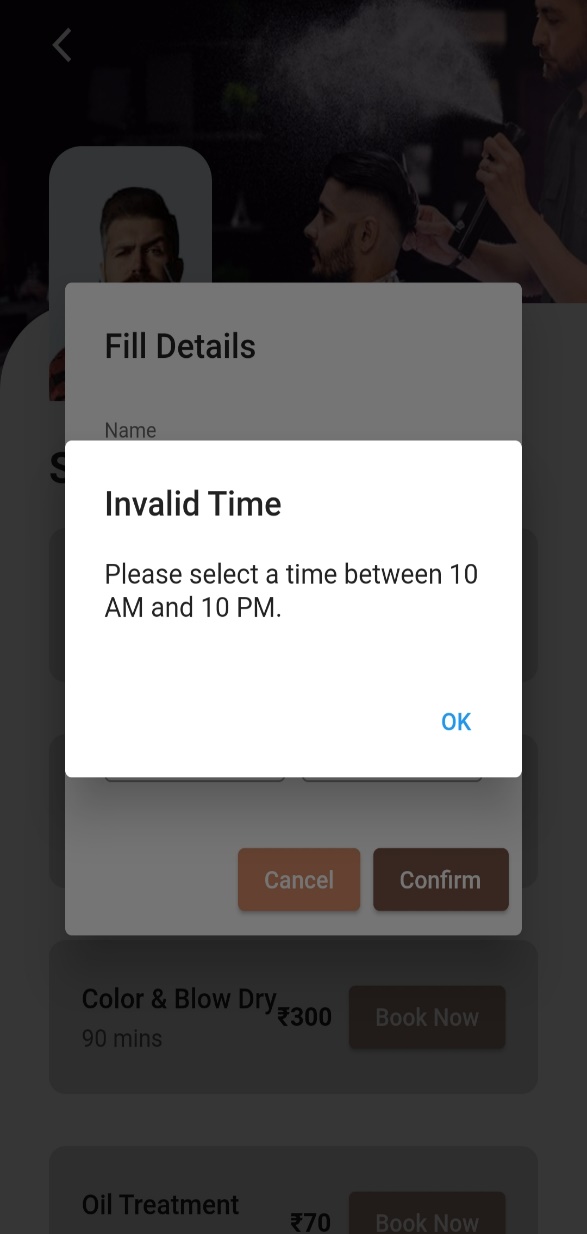
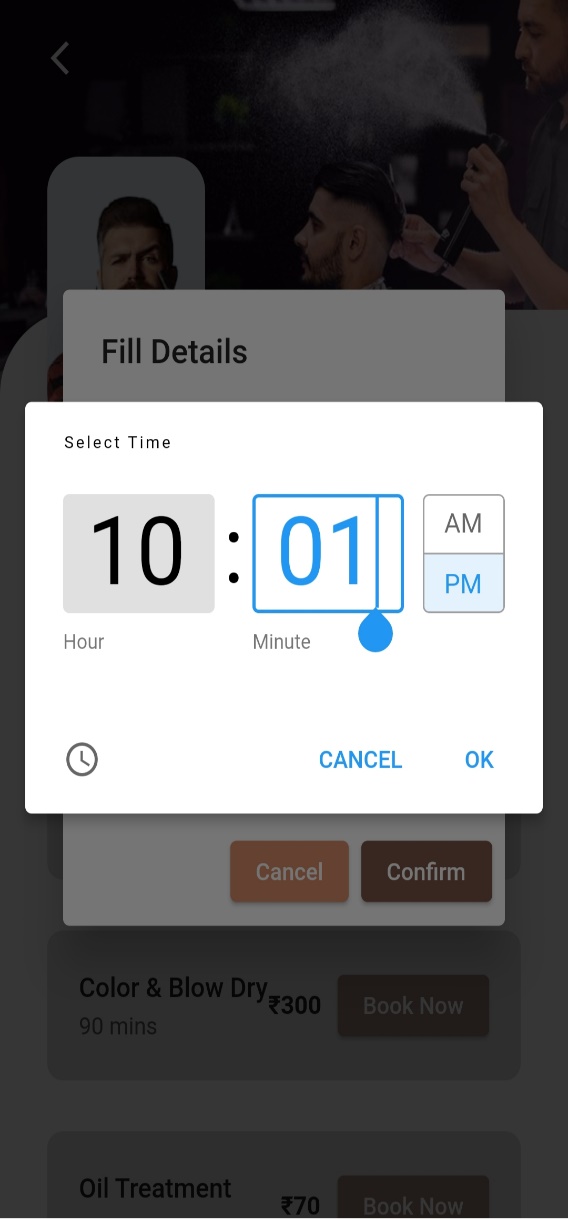
Output: Valid: Allow time to upload to database.

Invalid: Fail to upload time to database.

Result: Valid: Time between 10 AM and 10 PM is select to upload for

database.

**11.Conclusions**

In conclusion, our Men's Salon Booking App revolutionizes the way you schedule and manage appointments at our salon. With a seamless and user-friendly interface, both salon owners and customers can enjoy the benefits of digital appointment booking.

For salon owners, our app provides a powerful tool to effortlessly track appointments. Say goodbye to manual systems and embrace the efficiency of digital scheduling. You'll have full control over your salon's schedule, ensuring optimal utilization of your resources and delivering a smooth and enjoyable experience for your clients.

As a customer, you'll appreciate the convenience and flexibility our app offers. Browse through the available time slots and easily book an appointment that fits your schedule. Say goodbye to long waiting times and the frustration of trying to find an open slot. Our app ensures you can secure the desired time and date for your grooming needs.

We pride ourselves on our team of skilled stylists who are dedicated to providing top-notch services. With our app, you can trust that your appointment will be honored, allowing you to plan your day with confidence. Experience the luxury of a well-groomed appearance without the stress and hassle of traditional booking systems.

**12.Limitations Of The Project**

* The Application runs only on Android Platform with internet connection Enabled.
* The Application doesn’t support web based functionality.
* Admin can not alter the stylist details as of now.

**13.Future Scope And Further Enhancement Of The Project**

* The Application can be made responsive for all the platforms.
* Admin can add and remove stylist from the admin side.
* User can keep track of bookings.
* Admin and User both can get the notification reminder of the Appointments.

**14.Abbreviation And Acronyms**

**Document Reference**

Database Reference

A Firebase Reference represents a particular location in your database and can be used for reading or writing data to that Database location.

DataSnapshot

A DataSnapshot instance contains the data from a Firebase Database location.Any time you read data from the Database , you receive the data as a DataSnapshot.

getValue(): getValue() returns the data contained in the snapshot as ative types like Boolean,string,long,double,Map,List.

CFD : Context Flow Diagram

DFD : Data Flow Diagram

SDK : Software Development Kit

API : Application programming interface

JDK : Java Development Kit

IDE : Integrated Development Environment

**15.Bibliography**

* [www.stackoverflow.org](http://www.stackoverflow.org)
* <https://pub.dev/packages>