# MediSync - A Unified Platform for Secure Healthcare Management and Patient-Centric Services

*Project report submitted to the Amrita Vishwa Vidyapeetham in partial fulfillment of the requirement for the Degree of*

**BTech in Computer Science & Engineering**

*Submitted By :*

| | |
|---|---|
| Prabu CR | AM.ENU4CSE21344 |
| Rahulkrishnan Ravindran | AM.ENU4CSE21367 |
| Katari Sai Abhiram Varma | AM.ENU4CSE21376 |

*JULY 2024*

*AMRITA VISHWA VIDYAPEETHAM*

*AMRITA SCHOOL OF COMPUTING - ASC*

# Index

## Abstract

The envisioned healthcare application aims to revolutionize patient care by integrating seamless communication and comprehensive medical management features. At its core, the application prioritizes user privacy and security, utilizing AES 256 encryption to safeguard sensitive data. Each user is assigned a unique ID, ensuring secure and personalized interactions with healthcare providers.

Key functionalities include direct access to medical reports and prescriptions, empowering users to efficiently manage their health information from anywhere. Pharmacists can request prescriptions and deliver medications, while lab technicians and nurses can access and process user test results, facilitating prompt healthcare delivery.

In emergencies, an integrated button enables users to instantly share their location and details with hospitals, ensuring swift assistance when needed most. The application also supports insurance management, allowing users to input and manage insurance details for optimized coverage during medical services.

Appointment scheduling with doctors is simplified, enhancing convenience for users seeking healthcare consultations. A sophisticated chatbot, powered by ChatGPT 3.5 Turbo, offers intelligent responses and guidance, utilizing API tokenization for accuracy and reliability when immediate doctor availability isn't feasible.

Overall, the application aims to empower users by providing them with robust tools for managing their healthcare needs efficiently and securely, bridging the gap between medical facilities and patients through innovative technology and comprehensive service integration.

# Chapter One : Communication

**Client:**
The client for this healthcare application is the end user, typically patients seeking convenient access to medical records, prescriptions, and healthcare services. They utilize the application to manage personal health information securely and interact with healthcare providers.

**Minutes of Meeting:**

During the initial project meeting, stakeholders discussed the need for a comprehensive healthcare application that integrates various functionalities to enhance patient care. Key points included the requirement for secure data management using AES 256 encryption, direct access to medical reports and prescriptions, integration of emergency services, insurance management capabilities, and appointment scheduling. The meeting highlighted the importance of user privacy, efficient communication channels between users and healthcare providers, and the need for a user-friendly interface that supports seamless interaction.

Here the role of the stakeholder and developer was shuffled and alternatively selected by every member in the group in order to find newer solutions and  functionalities and features to implement in the application.

**Problem Statement:**
The current challenge lies in bridging the gap between patients and healthcare providers through a unified digital platform. Existing systems often lack robust security measures and fail to provide streamlined access to healthcare services. Patients face difficulties managing medical records, scheduling appointments, and accessing timely medical assistance during emergencies. There is a need for an integrated solution that ensures secure communication, enhances user experience, and facilitates efficient healthcare management from anywhere at any time.

# Chapter Two : Planning

**Deadlines:** The project is structured with several key deadlines to ensure timely completion of each phase:

- Phase 1: Design and Prototyping - Deadline: 1 week
- Phase 2: Development and Testing - Deadline: 1.5 weeks
- Phase 3: Deployment and Final Testing - Deadline: 1.5 weeks
  (Cumulative work / Overlapping phases)

**Phases:**

1. **Phase 1: Design and Prototyping**
    - Define application architecture and database schema.
    - Create wireframes and user interface designs.
    - Finalize encryption and security protocols.
2. **Phase 2: Development and Testing**
    - Implement frontend using Flutter SDK and backend using Firebase.
    - Integrate AES 256 encryption for data security.
    - Develop features including medical records access, prescription management, emergency button, insurance management, appointment scheduling, and chatbot functionality.
    - Conduct unit testing and integration testing to ensure functionality and security.

3. **Phase 3: Deployment and Final Testing**
    - Deploy the application on Firebase platform.
    - Conduct a final round of testing to verify performance, security, and user experience.
    - Prepare user documentation and support materials.
    - Plan for maintenance and updates post-deployment.

**Weekly Plans:** Throughout each phase, weekly plans are established to track progress and milestones:

- Conduct weekly sprint planning meetings to define tasks and goals.
- Assign tasks to developers and stakeholders.
- Review progress, address challenges, and adjust plans as needed.

**Budget:** The project operates on a null budget as resources like Firebase and Flutter SDK are available for free online. The focus is on leveraging open-source tools and maximizing efficiency without financial constraints.

**Roles:**

- **Scrum Master:** Rotates among team members to facilitate sprint planning, daily stand-ups, and retrospective meetings.
- **Stakeholders:** Involved in defining requirements, providing feedback, and ensuring alignment with project goals similar to the Scrum master role this role was also alternatively taken by every member in the group so that we had a wider spectrum of ideas to implement in the project.
- **Developers:** Responsible for implementing features, testing functionality, and contributing to continuous improvement of the application, every member in the group actively worked as a developer and was assigned certain task sets to complete throughout the completion of the project.

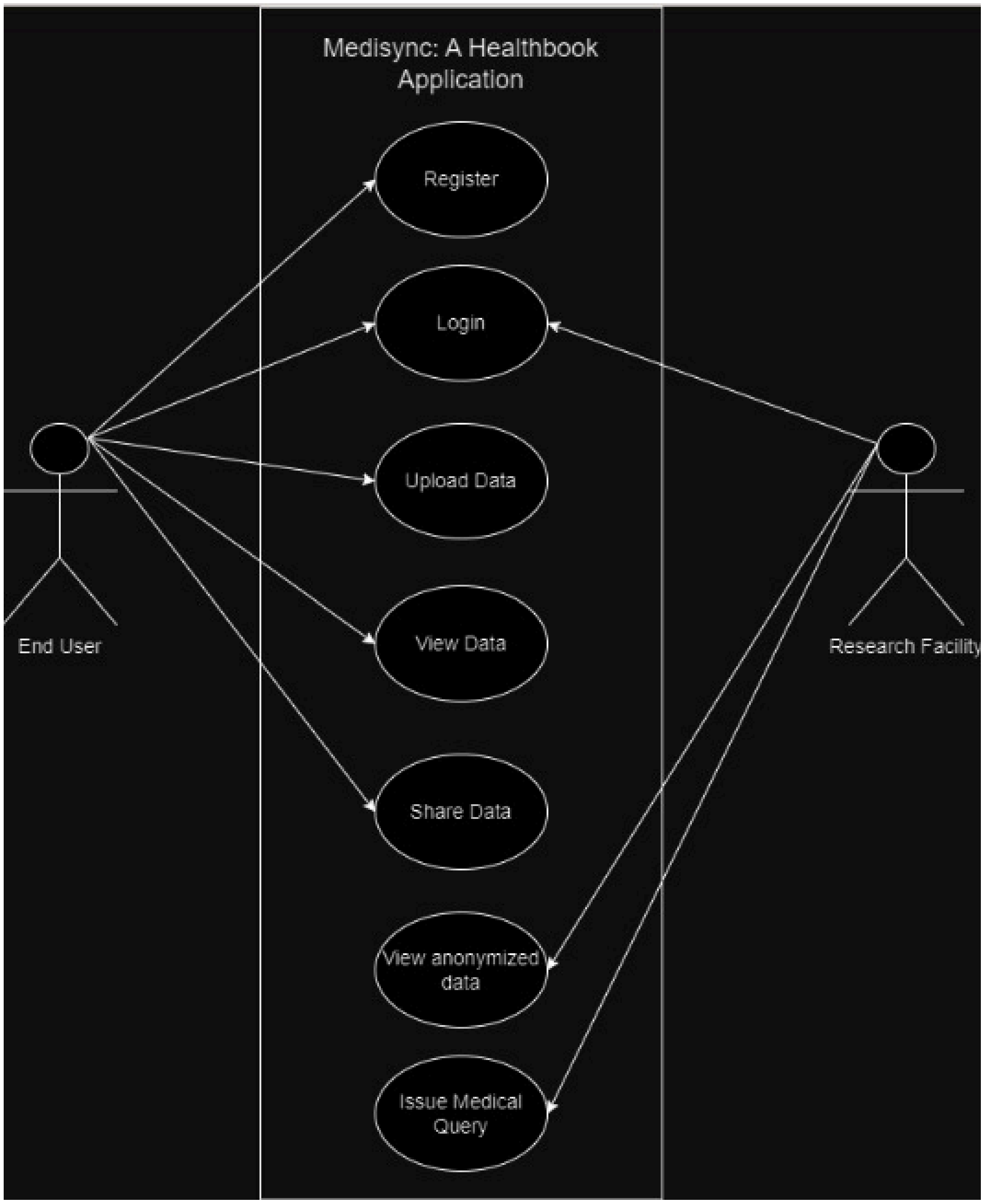
The iterative approach of alternating roles between stakeholders and developers encourages innovation and diverse perspectives in designing and implementing new functionalities for the healthcare application.

# Chapter Three : Modeling & Design Phase

The healthcare application undergoes meticulous modeling and design phases, focusing on structuring its architecture and functionalities for efficiency and clarity.

**Use Case Diagram:**

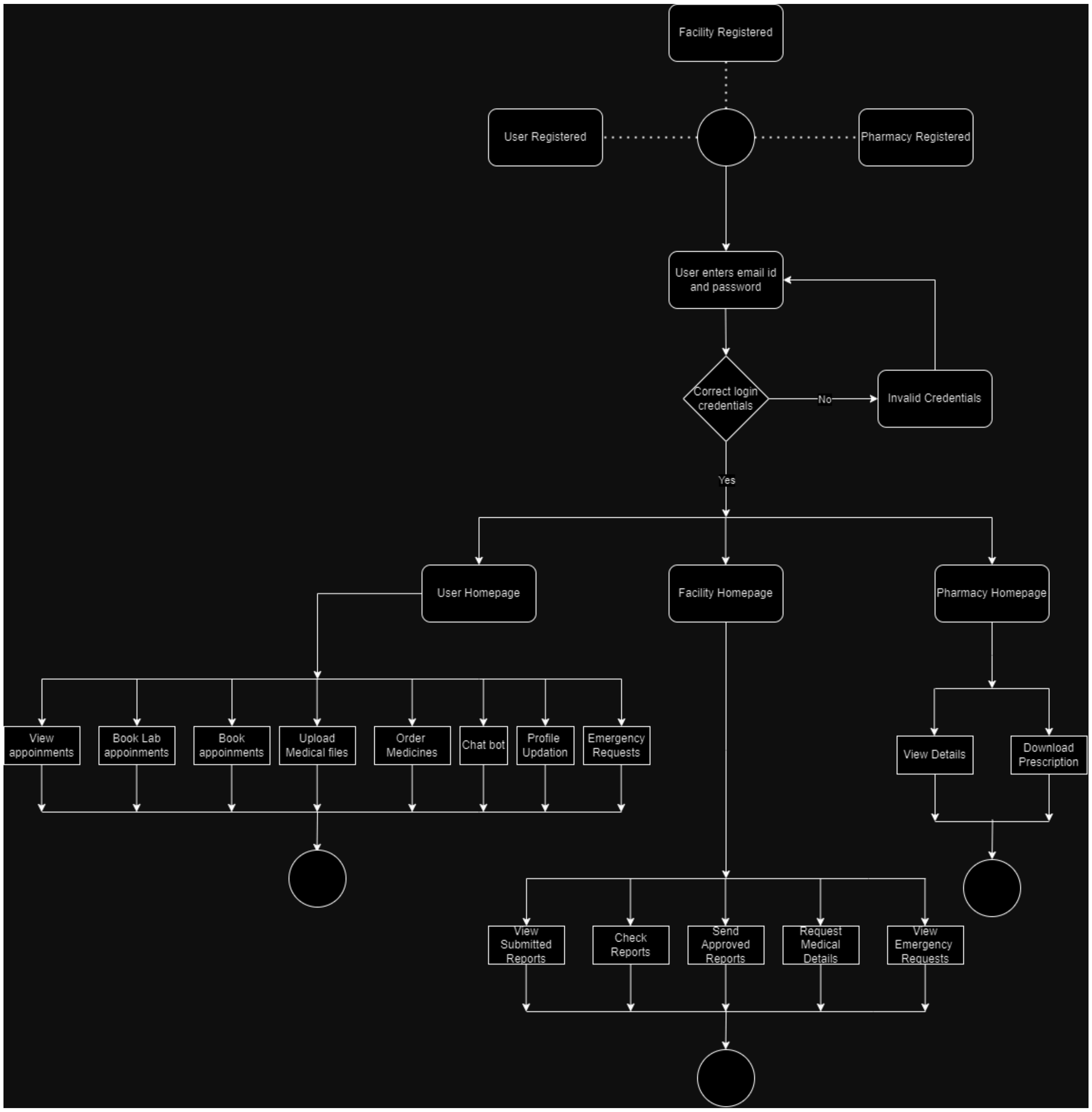
- Illustrates interactions between actors (users, doctors, pharmacists, etc.) and the system (healthcare application).



- Depicts key functionalities such as medical records access, emergency button activation, appointment scheduling, insurance management, and pharmacy interactions.

**Activity Diagram:**

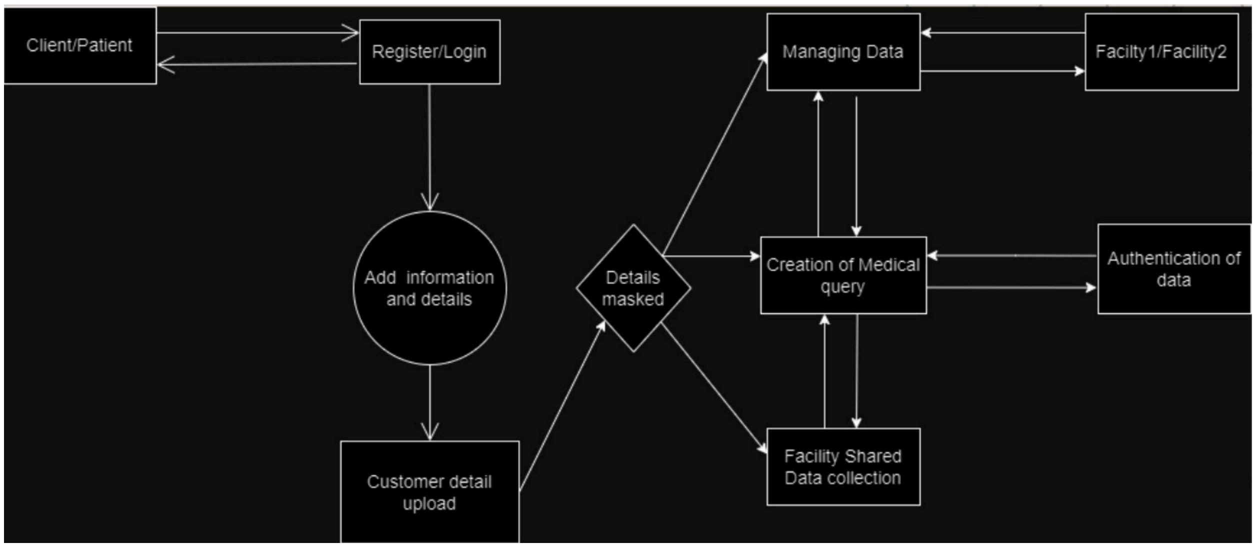
- Details workflows within specific features, showing step-by-step user interactions.



- Describes how users interact with the pharmacy module to request prescriptions, receive medication deliveries, and manage pharmacy-related tasks.

**Data Flow Diagram (DFD):**

- Visualizes data flow among components including user interfaces, backend services (Firebase Firestore), and external systems (e.g., pharmacy databases).



- Shows how data moves during prescription requests, medication deliveries, and updates between the application and pharmacy services.

**Modules:**

**1. Authentication Module:**

- Manages user authentication and session handling securely using Firebase Authentication.
- Ensures access control for protected features and data within the application.

**2. Medical Records Module:**

- Stores, retrieves, and manages user medical records securely in Firebase Firestore.
- Integrates AES 256 encryption to safeguard sensitive health information.

**3. Emergency Services Module:**

- Implements emergency features such as the emergency button for instant alerts to medical facilities.
- Utilizes real-time location tracking and notification capabilities using Firebase Cloud Functions.

**4. Appointment Scheduling Module:**

- Enables users to schedule, modify, and cancel appointments with healthcare providers.
- Sends notifications and updates via Firebase Cloud Messaging to users and healthcare providers.

**5.Insurance Management Module:**

- Allows users to input, update, and manage insurance details within the application.

- Validates insurance coverage and benefits through integration with external APIs or databases.

**6. Pharmacy Module:**

- Facilitates interactions between users and pharmacists/pharmacies.

- Enables users to request prescriptions, receive medication deliveries, and manage pharmacy-related tasks through the application.

- Integrates with pharmacy databases or APIs to verify prescriptions and track medication orders.

Each module is designed with modularity and scalability in mind, ensuring that the healthcare application can grow and adapt to meet user needs and technological advancements. The modeling and design phase lays a solid groundwork for subsequent implementation, testing, and deployment phases, ensuring a robust and user-friendly healthcare solution.

# Chapter 4 : Implementation & Code

The focus of the MediSync application shifts to the actual implementation of the planned features and functionalities, leveraging Firebase Firestore as a key component for backend data management.

**1. Backend Development with Firebase Firestore:**
- Utilize Firebase Firestore for real-time NoSQL database management, ensuring efficient storage and retrieval of user data such as medical records, prescriptions, and user profiles.
- Implement Firebase Authentication for secure user authentication and authorization, allowing users to securely access their personal health information.
- Integrate Firebase Cloud Functions to handle backend logic, such as data validation, business rules enforcement, and server-side computations.

**2. Frontend Development with Flutter SDK:**
- Develop the frontend using Flutter SDK to create a responsive and intuitive mobile application interface.
- Design user-friendly screens based on finalized wireframes and UI designs from Phase 1, ensuring a seamless user experience across iOS and Android platforms.
- Implement Firebase SDK in Flutter to establish real-time communication between the frontend application and Firestore backend.

**3. Feature Implementation:**

- *Medical Records Access:* Enable users to securely view, update, and manage their medical reports and prescriptions stored in Firestore.
- *Emergency Button:* Implement functionality for users to trigger emergency alerts, sending their location and details directly to medical facilities via Firestore Cloud Functions for immediate assistance.
- *Insurance Management:* Allow users to input and manage insurance details within the application, leveraging Firestore to store and retrieve insurance information securely.
- *Appointment Scheduling:* Integrate a feature for users to schedule appointments with healthcare providers, with Firestore managing appointment data and notifications.
- *Chatbot Integration:* Introduce a ChatGPT-powered chatbot feature, utilizing Firestore to store conversation histories securely and deliver accurate medical information.

**4. Testing and Quality Assurance:**

- Conduct rigorous testing throughout development, including unit testing and integration testing to ensure each feature functions correctly and securely with Firestore integration.
- Perform user acceptance testing (UAT) to validate that the application meets user expectations and operates smoothly under various scenarios.

**5. Documentation, Training, Deployment, and Support:**

- Prepare comprehensive documentation detailing application features, usage guidelines, and troubleshooting tips for users and stakeholders.
- Conduct training sessions to familiarize stakeholders with the application and its capabilities, including Firebase Firestore integration.
- Deploy the application to Firebase Hosting, ensuring scalability and reliability with Firestore as the backend database.
- Provide ongoing maintenance and support post-launch, monitoring application performance, addressing user feedback, and implementing updates as necessary.

This chapter represents the pivotal phase where theoretical planning transitions into practical implementation, utilizing Firebase Firestore to enhance data management, security, and real-time functionality within the healthcare application.

## Code:

### App_page.dart

```dart
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class ViewAppointmentsPage extends StatelessWidget {
  const ViewAppointmentsPage({super.key});

  @override
  Widget build(BuildContext context) {
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;

    return Scaffold(
      appBar: AppBar(
        title: const Text('View Appointments'),
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream: _firestore.collection('appointments').snapshots(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
            return const Center(child: Text('No appointments found'));
          }
          return ListView(
            children: snapshot.data!.docs.map((doc) {
              var data = doc.data() as Map<String, dynamic>;
              return ListTile(
                title: Text('${data['date']} - ${data['time']}'),
                subtitle: Text(
                    '${data['specialization']} - Dr.
${data['doctorName']}'),
              );
            }).toList(),
          );
        },
      ),
    );
  }
}
```

**Book_appointment.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class BookDoctorAppointmentPage extends StatefulWidget {
  const BookDoctorAppointmentPage({Key? key}) : super(key: key);

  @override
  _BookDoctorAppointmentPageState createState() =>
_BookDoctorAppointmentPageState();
}

class _BookDoctorAppointmentPageState extends
State<BookDoctorAppointmentPage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final TextEditingController dateController = TextEditingController();
  final TextEditingController doctorNameController =
TextEditingController();
  String specialization = 'ENT';
  String time = '9:00 AM';

  Future<void> _bookAppointment() async {
    User? user = _auth.currentUser;
    if (user != null) {
      await _firestore.collection('appointments').add({
        'userId': user.uid,
        'date': dateController.text,
        'time': time,
        'specialization': specialization,
        'doctorName': doctorNameController.text,
      });
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Appointment booked successfully')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Book Doctor Appointment'),
        backgroundColor: Colors.teal,
      ),
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: ListView(
              children: <Widget>[
                TextField(
                  controller: dateController,
                  decoration: const InputDecoration(
                          labelText: 'Date',
                          border: OutlineInputBorder(),
                          filled: true,
                          fillColor: Colors.white,
                        ),
                  onTap: () async {
                    DateTime? pickedDate = await showDatePicker(
                      context: context,
                      initialDate: DateTime.now(),
                      firstDate: DateTime(2020),
                      lastDate: DateTime(2025),
                    );
                    if (pickedDate != null) {
                      dateController.text =
pickedDate.toString().substring(0, 10);
```

```
                    }
                  },
                ),
                const SizedBox(height: 10),
                DropdownButtonFormField<String>(
                  decoration: const InputDecoration(
                    labelText: 'Time',
                    border: OutlineInputBorder(),
                    filled: true,
                    fillColor: Colors.white,
                  ),
                  value: time,
                  onChanged: (String? newValue) {
                    setState(() {
                      time = newValue!;
                    });
                  },
                  items: <String>[
                    '9:00 AM', '10:00 AM', '11:00 AM', '12:00 PM', '4:00
PM', '5:00 PM', '6:00 PM', '7:00 PM', '8:00 PM'
                  ].map<DropdownMenuItem<String>>((String value) {
                    return DropdownMenuItem<String>(
                      value: value,
                      child: Text(value),
                    );
                  }).toList(),
                ),
                const SizedBox(height: 10),
                DropdownButtonFormField<String>(
                  decoration: const InputDecoration(
                    labelText: 'Specialization',
                    border: OutlineInputBorder(),
                    filled: true,
                    fillColor: Colors.white,
                  ),
                  value: specialization,
                  onChanged: (String? newValue) {
                    setState(() {
                      specialization = newValue!;
                    });
                  },
                  items: <String>['ENT', 'Ortho', 'Gastro', 'Cardio',
'Neuro', 'Dentist']
                      .map<DropdownMenuItem<String>>((String value) {
                    return DropdownMenuItem<String>(
                      value: value,
                      child: Text(value),
                    );
                  }).toList(),
                ),
                const SizedBox(height: 10),
                TextFormField(
                  controller: doctorNameController,
                  decoration: const InputDecoration(
                    labelText: 'Doctor Name',
                    border: OutlineInputBorder(),
                    filled: true,
                    fillColor: Colors.white,
                  ),
                ),

                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: _bookAppointment,
                  child: const Text('Book Appointment'),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.teal,
                    foregroundColor: Colors.white,
                    minimumSize: const Size(double.infinity, 50),
                    shadowColor: Colors.transparent,
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

**Book_lab_test.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class BookLabTestPage extends StatefulWidget {
  const BookLabTestPage({Key? key}) : super(key: key);

  @override
  _BookLabTestPageState createState() => _BookLabTestPageState();
}

class _BookLabTestPageState extends State<BookLabTestPage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final TextEditingController dateController = TextEditingController();
  String testType = 'Blood Test';

  Future<void> _bookLabTest() async {
    User? user = _auth.currentUser;
    if (user != null) {
      await _firestore.collection('labTests').add({
        'userId': user.uid,
        'date': dateController.text,
        'testType': testType,
      });
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Lab test booked successfully')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Book Lab Test'),
        backgroundColor: Colors.teal,
      ),
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: ListView(
              children: <Widget>[
                TextField(
                  controller: dateController,
                  decoration: const InputDecoration(
                          labelText: 'Date',
                          border: OutlineInputBorder(),
                          filled: true,
                          fillColor: Colors.white,
                        ),
                  onTap: () async {
                    DateTime? pickedDate = await showDatePicker(
                      context: context,
                      initialDate: DateTime.now(),
                      firstDate: DateTime(2020),
                      lastDate: DateTime(2025),
                    );
                    if (pickedDate != null) {
                      dateController.text =
pickedDate.toString().substring(0, 10);
                    }
                  },
                ),
                const SizedBox(height: 10),
                DropdownButtonFormField<String>(
                  decoration: const InputDecoration(
                    labelText: 'Specialization',
                    border: OutlineInputBorder(),
                    filled: true,
                    fillColor: Colors.white,
                  ),
                  value: testType,
                  onChanged: (String? newValue) {
                    setState(() {
                      testType = newValue!;
                    });
                  },
                  items: <String>['Blood Test', 'X-Ray', 'MRI', 'CT Scan',
'Ultrasound', 'ECG']
                      .map<DropdownMenuItem<String>>((String value) {
                    return DropdownMenuItem<String>(
                      value: value,
                      child: Text(value),
                    );
                  }).toList(),
                ),
                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: _bookLabTest,
                  child: const Text('Book Lab Test'),
                  style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
```

```
  }
}
```

**Check_reports.dart**

```dart
import 'package:flutter/material.dart';

class CheckReportsPage extends StatelessWidget {
  const CheckReportsPage({super.key});

  @override
  Widget build(BuildContext context) {
    // Add your code to check and approve reports
    return Scaffold(
      appBar: AppBar(
        title: const Text('Check Reports'),
      ),
      body: Center(
        child: const Text('Here you can check and approve reports.'),
      ),
    );
  }
}
```

**Facility_page.dart**

```dart
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'profile_page.dart';
import 'view_submitted_reports.dart';
import 'check_reports.dart';
import 'send_approved_reports.dart';
import 'request_medical_details_page.dart'; // Import the new page

class FacilityPage extends StatefulWidget {
  const FacilityPage({Key? key}) : super(key: key);
  @override
  _FacilityPageState createState() => _FacilityPageState();
}

class _FacilityPageState extends State<FacilityPage> {
  bool hasEmergencyRequest = false; // Track emergency request status
  List<Map<String, dynamic>> emergencyRequests = []; // Store emergency
requests

  @override
  void initState() {
    super.initState();
    _checkEmergencyRequest();
    _fetchEmergencyRequests();
  }

  Future<void> _checkEmergencyRequest() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    bool? emergencyRequest = prefs.getBool('emergencyRequest') ?? false;
    setState(() {
      hasEmergencyRequest = emergencyRequest;
    });
  }

  Future<void> _fetchEmergencyRequests() async {

FirebaseFirestore.instance.collection('emergency_requests').snapshots().li
sten((snapshot) async {
      List<Map<String, dynamic>> requests = [];
      for (var doc in snapshot.docs) {
        Map<String, dynamic> requestData = doc.data() as Map<String,
dynamic>;
        DocumentSnapshot userSnapshot = await
FirebaseFirestore.instance.collection('users').doc(requestData['userId']).
get();
        if (userSnapshot.exists) {
          requestData['userName'] = userSnapshot['name'];
        } else {
          requestData['userName'] = 'Unknown';
        }
        requests.add(requestData);
      }
      setState(() {
        emergencyRequests = requests;
      });
    });
  }

  void _showEmergencyRequests() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Emergency Requests'),
        content: emergencyRequests.isEmpty
            ? const Text('No emergency requests found.')
            : Column(
                mainAxisSize: MainAxisSize.min,
                children: emergencyRequests.map((request) {
                  return ListTile(
                    title: Text('User: ${request['userName']}'),
                    subtitle: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text('Timestamp:
${request['timestamp'].toDate().toString()}'),
                        Text('Locality: ${request['locality'] ??
'Unknown'}'),
                      ],
                    ),
                  );
                }).toList(),
              ),
        actions: <Widget>[
          TextButton(
            child: const Text('OK'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}

  @override
  Widget build(BuildContext context) {
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;

    return Scaffold(
      appBar: PreferredSize(
        preferredSize: Size.fromHeight(0.0), // Hide the app bar
        child: Container(),
      ),
      body: Column(
        children: [
          Expanded(
            child: StreamBuilder<QuerySnapshot>(
              stream: _firestore.collection('appointments').snapshots(),
              builder: (context, snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                }
                if (snapshot.hasError) {
                  return Center(child: Text('Error: ${snapshot.error}'));
                }
                if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
                  return const Center(child: Text('No appointments
found'));
                }
                return SingleChildScrollView(
                  scrollDirection: Axis.horizontal,
                  child: DataTable(
                    columns: const [
                      DataColumn(label: Text('Date')),
                      DataColumn(label: Text('Time')),
                      DataColumn(label: Text('Specialization')),
                      DataColumn(label: Text('Doctor Name')),
                      DataColumn(label: Text('User ID')),
                    ],
                    rows: snapshot.data!.docs.map((doc) {
                      var data = doc.data() as Map<String, dynamic>;
                      return DataRow(
                        cells: [
                          DataCell(Text(data['date'])),
                          DataCell(Text(data['time'])),
                          DataCell(Text(data['specialization'])),
                          DataCell(Text('Dr. ${data['doctorName']}')),
                          DataCell(Text(data['userId'])),
                        ],
                      );
                    }).toList(),
                  ),
                );
              },
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              children: <Widget>[
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => const
ViewSubmittedReportsPage()),
                    );
                  },
                  child: const Text('View Submitted Reports'),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => const
CheckReportsPage()),
                    );
                  },
                  child: const Text('Check Reports'),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => const
SendApprovedReportsPage()),
                    );
                  },
                  child: const Text('Send Approved Reports'),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(builder: (context) => const
RequestMedicalDetailsPage()), // Navigate to the new page
                    );
                  },
                  child: const Text('Request Medical Details'),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  onPressed: () {
                    _showEmergencyRequests(); // Show emergency requests
                  },
                  child: const Text('Emergency Requests'),
                ),
                const SizedBox(height: 10),
                BottomAppBar(
                  child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      TextButton(
                        onPressed: () async {
                          SharedPreferences prefs = await
SharedPreferences.getInstance();
                          await FirebaseAuth.instance.signOut();
                          prefs.remove('lastPage');

Navigator.of(context).pushNamedAndRemoveUntil('/', (route) => false);
                        },
                        child: const Text('Logout'),
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

**Firebase_options.dart**

```dart
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      return web;
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        return macos;
      case TargetPlatform.windows:
        return windows;
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI
again.',
        );
      default:
        throw UnsupportedError(
```

```
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyA134EU1xdlycxqvN35dEHrTcMPM8TbBBM',
  appId: '1:360324002588:web:7ca7e78d8870fcb15d40d9',
  messagingSenderId: '360324002588',
  projectId: 'medi-74002',
  authDomain: 'medi-74002.firebaseapp.com',
  storageBucket: 'medi-74002.appspot.com',
  measurementId: 'G-73ZZRMVPE4',
);

static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyADVWxLXLhwhhJvhgas0Q2dKuQE4mKom04',
  appId: '1:360324002588:android:1316d0216171602e5d40d9',
  messagingSenderId: '360324002588',
  projectId: 'medi-74002',
  storageBucket: 'medi-74002.appspot.com',
);

static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyAWnr34ekMkqU3Z2lUE7fnOI20fbhZ55NA',
  appId: '1:360324002588:ios:346369efeac8581a5d40d9',
  messagingSenderId: '360324002588',
  projectId: 'medi-74002',
  storageBucket: 'medi-74002.appspot.com',
  iosBundleId: 'com.example.ex',
);

static const FirebaseOptions macos = FirebaseOptions(
  apiKey: 'AIzaSyAWnr34ekMkqU3Z2lUE7fnOI20fbhZ55NA',
  appId: '1:360324002588:ios:346369efeac8581a5d40d9',
  messagingSenderId: '360324002588',
  projectId: 'medi-74002',
  storageBucket: 'medi-74002.appspot.com',
  iosBundleId: 'com.example.ex',
);
```

```
  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyA134EU1xdlycxqvN35dEHrTcMPM8TbBBM',
    appId: '1:360324002588:web:dc1e0c7d9c3ced595d40d9',
    messagingSenderId: '360324002588',
    projectId: 'medi-74002',
    authDomain: 'medi-74002.firebaseapp.com',
    storageBucket: 'medi-74002.appspot.com',
    measurementId: 'G-QWECZG3YRS',
  );
}
```

**Firebase_storage_service.dart**

```
import 'package:firebase_storage/firebase_storage.dart';

class FirebaseStorageService {
  final FirebaseStorage _storage = FirebaseStorage.instance;

  Future<String?> getFileUrl(String filePath) async {
    try {
      final ref = _storage.ref().child(filePath);
      final downloadUrl = await ref.getDownloadURL();
      // Append a dummy query parameter to the download URL
      // This makes the URL unique but does not affect the file retrieval
      final viewUrl = '$downloadUrl?view=true';
      return viewUrl;
    } catch (e) {
      print('Error getting file URL: $e');
      return null;
    }
  }
}
```

**Home_page.dart**

```
import 'package:ex/app_page.dart';
import 'package:ex/book_appointment.dart';
```

```dart
import 'package:ex/book_lab_test.dart';
import 'package:ex/order_medicine.dart';
import 'package:ex/profile_page.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:file_picker/file_picker.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

class HomePage extends StatelessWidget {
  const HomePage({Key? key});

  Future<void> _setLastVisitedPage(BuildContext context) async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setString('lastPage', 'HomePage');
  }

  Future<void> _pickFile(BuildContext context) async {
    FilePickerResult? result = await FilePicker.platform.pickFiles();

    if (result != null) {
      PlatformFile file = result.files.first;

      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text('File Selected'),
            content: Text('File: ${file.name}'),
            actions: <Widget>[
              TextButton(
                child: const Text('OK'),
                onPressed: () {
                  Navigator.of(context).pop();
                },
              ),
            ],
          );
        },
      );
    } else {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text('No File Selected'),
            content: const Text('You did not select any file.'),
            actions: <Widget>[
              TextButton(
                child: const Text('OK'),
                onPressed: () {
                  Navigator.of(context).pop();
                },
              ),
            ],
          );
        },
      );
    }
  }

  Future<void> _openChatBot(BuildContext context) async {
    TextEditingController _controller = TextEditingController();
    String? responseText;

    showDialog(
      context: context,
      builder: (BuildContext context) {
        return StatefulBuilder(
          builder: (context, setState) {
            return AlertDialog(
              title: const Text('ChatBot'),
              content: Column(
                mainAxisSize: MainAxisSize.min,
                children: [
                  TextField(
                    controller: _controller,
                    decoration: const InputDecoration(
                      hintText: 'Enter your query',
                    ),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    onPressed: () async {
                      String query = _controller.text;
                      if (query.isNotEmpty) {
                        // Call the ChatGPT API
                        String response = await
fetchChatGptResponse(query);
                        setState(() {
                          responseText = response;
                          _controller.clear();
                        });
                      }
                    },
                    child: const Text('Send'),
                  ),
                  const SizedBox(height: 10),
                  if (responseText != null)
                    Text(
                      responseText!,
                      style: const TextStyle(fontStyle: FontStyle.italic),
                    ),
                ],
              ),
              actions: <Widget>[
                TextButton(
                  child: const Text('Close'),
                  onPressed: () {
                    Navigator.of(context).pop();
                  },
                ),
              ],
            );
          },
        );
      },
    );
  }
```

```dart
  Future<String> fetchChatGptResponse(String query) async {
    final url = Uri.parse('https://api.openai.com/v1/chat/completions');
    final response = await http.post(
      url,
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer
sk-proj-e67UrvskcuCTxOis5cbDT3BlbkFJUaq1yHZcCZ15WyuDzOyB', // Replace with
your actual API key
      },
      body: json.encode({
        'model': 'gpt-3.5-turbo',
        'messages': [
          {'role': 'system', 'content': 'You are a helpful assistant.'},
          {'role': 'user', 'content': query},
        ],
        'max_tokens': 150,
      }),
    );

    if (response.statusCode == 200) {
      print('Response body: ${response.body}');
      final jsonResponse = json.decode(response.body);
      return jsonResponse['choices'][0]['message']['content'].trim();
    } else {
      print('Request failed with status: ${response.statusCode}');
      print('Response body: ${response.body}');
      return 'Failed to get response from MediSync';
    }
  }

  @override
  Widget build(BuildContext context) {
    _setLastVisitedPage(context);

    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: const Text('MediSync Home'),
        backgroundColor: Colors.teal,
        actions: <Widget>[
          IconButton(
            icon: const Icon(Icons.account_circle),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => const
ProfilePage()),
              );
            },
          ),
        ],
      ),
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Ensure
this path is correct
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Center(
            child: SingleChildScrollView(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  const SizedBox(height: 100),
                  const Text(
                    'Welcome to MediSync!',
                    style: TextStyle(fontSize: 36, fontWeight:
FontWeight.bold),
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
BookDoctorAppointmentPage()),
                      );
                    },
                    child: const Text('Book Doctor Appointment'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
BookLabTestPage()),
                      );
                    },
                    child: const Text('Book Lab Test'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
ViewAppointmentsPage()),
                      );
                    },
                    child: const Text('View Appointments'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
OrderMedicinePage()),
                      );
                    },
                    child: const Text('Order Medicines'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      _pickFile(context);
                    },
                    child: const Text('Upload Health Records'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: const Text('Request Received'),
                            content: const Text('Your details have been
sent to the facility. We will get back to you shortly.'),
                            actions: <Widget>[
                              TextButton(
                                child: const Text('OK'),
                                onPressed: () {
                                  Navigator.of(context).pop();
                                },
                              ),
                            ],
                          );
                        },
                      );
                    },
                    child: const Text('Emergency Alert'),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30),
                      ),
                    ),
                    onPressed: () {
                      _openChatBot(context);
                    },
                    child: const Text('Chat with MediSync'),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

**Image_display.dart**

```
import 'package:flutter/material.dart';

class ImageDisplayPage extends StatelessWidget {
  final String imageUrl;

  const ImageDisplayPage({Key? key, required this.imageUrl}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Image Display'),
      ),
      body: Center(
        child: Image.network(
          imageUrl,
          loadingBuilder: (BuildContext context, Widget child,
ImageChunkEvent? loadingProgress) {
            if (loadingProgress == null) {
              return child;
            }
            return Center(
              child: CircularProgressIndicator(
                value: loadingProgress.expectedTotalBytes != null
                    ? loadingProgress.cumulativeBytesLoaded /
loadingProgress.expectedTotalBytes!
                    : null,
              ),
            );
          },
          errorBuilder: (BuildContext context, Object exception,
StackTrace? stackTrace) {
            print('Failed to load image: $exception'); // Debugging line
            return const Text('Failed to load image');
          },
        ),
      ),
      );
   }
}
```

**Launchpage.dart**

```dart
import 'package:flutter/material.dart';
import 'package:ex/login_page.dart';
import 'package:ex/register_page.dart';

class LaunchPage extends StatelessWidget {
  const LaunchPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Center(
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center, // Center the
content
                children: <Widget>[
                  const Text(
                    'Welcome to MediSync',
                    style: TextStyle(
                      fontSize: 28,
                      fontWeight: FontWeight.bold,
                      color: Colors.black,
                      shadows: [
                        Shadow(
                          blurRadius: 10.0,
                          color: Colors.black,
                          offset: Offset(5.0, 5.0),
                        ),
                      ],
                    ),
                    textAlign: TextAlign.center,
                  ),
                  const SizedBox(height: 40), // Add space between text
and buttons
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal, // Changed to teal
                      foregroundColor: Colors.white, // Text color
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30), //
Capsule shape
                      ),
                      elevation: 5,
                      shadowColor: Colors.tealAccent,
                    ),
                    onPressed: () {
                      print('Login button pressed');
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
LoginPage()),
                      );
                    },
                    child: const Text(
                      'Login',
                      style: TextStyle(fontSize: 18),
                    ),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal, // Changed to teal
```

```
                      foregroundColor: Colors.white, // Text color
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30), //
Capsule shape
                      ),
                      elevation: 5,
                      shadowColor: Colors.tealAccent,
                    ),
                    onPressed: () {
                      print('Register button pressed');
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => const
RegisterPage()),
                      );
                    },
                    child: const Text(
                      'Register',
                      style: TextStyle(fontSize: 18),
                    ),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

## Login_page.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:ex/home_page.dart';
import 'package:ex/facility_page.dart';
import 'package:ex/pharmacy_page.dart'; // Import the PharmacyPage if it
exists

class LoginPage extends StatefulWidget {
  const LoginPage({Key? key});

  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController =
TextEditingController();
  final FirebaseAuth _auth = FirebaseAuth.instance;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Center(
            child: SingleChildScrollView(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  const SizedBox(height: 100),
                  const Text(
                    'Hi There,',
                    style: TextStyle(fontSize: 36, fontWeight:
FontWeight.bold),
                  ),
                  const Text(
                    'Enter your login credentials',
                    style: TextStyle(fontSize: 24),
                  ),
                  const SizedBox(height: 20),
                  Padding(
                    padding: const EdgeInsets.symmetric(horizontal: 16),
                    child: TextField(
                      controller: emailController,
                      decoration: InputDecoration(
                        labelText: 'Email',
                        border: OutlineInputBorder(),
                        filled: true,
                        fillColor: Colors.white,
                      ),
                    ),
                  ),
                  const SizedBox(height: 10),
                  Padding(
                    padding: const EdgeInsets.symmetric(horizontal: 16),
                    child: TextField(
                      controller: passwordController,
                      decoration: InputDecoration(
                        labelText: 'Password',
                        border: OutlineInputBorder(),
                        filled: true,
                        fillColor: Colors.white,
                      ),
                      obscureText: true,
                    ),
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal, // Changed to teal
                      foregroundColor: Colors.white, // Text color
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30), //
Capsule shape
                      ),
                    ),
                    onPressed: () async {
                      try {
                        UserCredential userCredential = await
_auth.signInWithEmailAndPassword(
                          email: emailController.text,
                          password: passwordController.text,
                        );
                        print('Login successful');
                        if (emailController.text.endsWith('@facility.in'))
{
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(builder: (context) => const
FacilityPage()),  // Navigate to facility page
                          );
                        } else if
(emailController.text.endsWith('@pharmacy.in')) {
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(builder: (context) => const
PharmacyPage()),  // Navigate to pharmacy page
                          );
                        } else {
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(builder: (context) => const
HomePage()),  // Navigate to home page
```

```
                        );
                      }
                    } catch (e) {
                      print('Login failed: $e');

ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                        content: Text('Login failed: $e'),
                      ));
                    }
                  },
                  child: const Text('Login'),
                ),
                const SizedBox(height: 10),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.teal, // Changed to teal
                    foregroundColor: Colors.white, // Text color
                    padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(30), //
Capsule shape
                    ),
                  ),
                  onPressed: () {
                    Navigator.pop(context); // Go back to the previous
page
                  },
                  child: const Text('Back'),
                ),
              ],
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```

**Main.dart**

```dart
import 'package:ex/facility_page.dart';
import 'package:ex/pharmacy_page.dart';
import 'package:ex/profile_page.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:ex/home_page.dart';
import 'package:ex/launchpage.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  try {
    await Firebase.initializeApp(
      options: FirebaseOptions(
        apiKey: "AIzaSyA134EU1xdlycxqvN35dEHrTcMPM8TbBBM",
        authDomain: "medi-74002.firebaseapp.com",
        projectId: "medi-74002",
        storageBucket: "medi-74002.appspot.com",
        messagingSenderId: "360324002588",
        appId: "1:360324002588:web:7ca7e78d8870fcb15d40d9",
        measurementId: "G-73ZZRMVPE4",
      ),
    );
    print('Firebase initialized successfully');
  } catch (e) {
    print('Error initializing Firebase: $e');
  }

  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'MediSync',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const SplashScreen(),
    );
  }
}

class SplashScreen extends StatefulWidget {
  const SplashScreen({super.key});

  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    _checkLastVisitedPage();
  }

  Future<void> _checkLastVisitedPage() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String? lastPage = prefs.getString('lastPage');


    if (lastPage == 'HomePage') {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const HomePage()),
      );
    } else if (lastPage == 'PharmacyPage') {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const PharmacyPage()),
      );
    } else if (lastPage == 'FacilityPage') {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const FacilityPage()),
      );
    } else {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const LaunchPage()),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: CircularProgressIndicator(),
      ),
    );
  }
}
```

**Order_medicine.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:file_picker/file_picker.dart';

class OrderMedicinePage extends StatefulWidget {
  const OrderMedicinePage({Key? key}) : super(key: key);

  @override
  _OrderMedicinePageState createState() => _OrderMedicinePageState();
}

class _OrderMedicinePageState extends State<OrderMedicinePage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final FirebaseStorage _storage = FirebaseStorage.instance;
  PlatformFile? pickedFile;

  Future<void> _orderMedicine() async {
    User? user = _auth.currentUser;
    if (user != null && pickedFile != null) {
      try {
        // Upload the file to Firebase Storage
        String fileName = pickedFile!.name;
        Reference storageRef =
_storage.ref().child('prescriptions/$fileName');
        UploadTask uploadTask = storageRef.putData(pickedFile!.bytes!);

        // Wait for the upload to complete
        TaskSnapshot snapshot = await uploadTask;
        String downloadUrl = await snapshot.ref.getDownloadURL();

        // Save the file metadata and URL to Firestore
        await _firestore.collection('medicines').add({
          'userId': user.uid,
          'fileName': fileName,
          'fileSize': pickedFile!.size,
          'fileUrl': downloadUrl,
        });

        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('Medicine order placed
successfully')),
        );
        Navigator.pop(context); // Navigate back to the previous page
      } catch (e) {
        print('Error uploading file: $e');
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Error uploading file: $e')),
        );
      }
    }
  }

  Future<void> _pickFile() async {
    FilePickerResult? result = await FilePicker.platform.pickFiles();
    if (result != null) {
      setState(() {
        pickedFile = result.files.first;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Order Medicine'),
        backgroundColor: Colors.teal,
      ),
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
```

```
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: ListView(
              children: <Widget>[
                ElevatedButton(
                  onPressed: _pickFile,
                  child: const Text('Upload Prescription'),
                  style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                ),
                if (pickedFile != null) Text('Selected file:
${pickedFile!.name}'),
                const SizedBox(height: 20),
                ElevatedButton(
                  onPressed: _orderMedicine,
                  child: const Text('Place Order'),
                  style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

**Pharmacy_page.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:url_launcher/url_launcher.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:firebase_auth/firebase_auth.dart';

class PharmacyPage extends StatefulWidget {
  const PharmacyPage({Key? key}) : super(key: key);

  @override
  _PharmacyPageState createState() => _PharmacyPageState();
}

class _PharmacyPageState extends State<PharmacyPage> {
  @override
  void initState() {
    super.initState();
    _setLastVisitedPage();
  }

  Future<void> _setLastVisitedPage() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setString('lastPage', 'PharmacyPage');
  }

  @override
  Widget build(BuildContext context) {
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;

    return Scaffold(
      appBar: AppBar(
        title: const Text('Pharmacy Orders'),
      ),
      body: StreamBuilder<QuerySnapshot>(
        stream: _firestore.collection('medicines').snapshots(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
            return const Center(child: Text('No orders found'));
          }
          return ListView(
            children: snapshot.data!.docs.map((doc) {
              var data = doc.data() as Map<String, dynamic>;
              return FutureBuilder<DocumentSnapshot>(
                future:
_firestore.collection('users').doc(data['userId']).get(),
                builder: (context, userSnapshot) {
                  if (userSnapshot.connectionState ==
ConnectionState.waiting) {
                    return const ListTile(
                      title: Text('Loading user info...'),
                    );
                  }
                  if (userSnapshot.hasError) {
                    return ListTile(
                      title: Text('Error: ${userSnapshot.error}'),
                    );
                  }
                  if (!userSnapshot.hasData || !userSnapshot.data!.exists)
{
                    return const ListTile(
                      title: Text('User not found'),
                    );
                  }

                  var userData = userSnapshot.data!.data() as Map<String,
dynamic>;
                  return ListTile(
                    title: Text('Prescription: ${data['fileName']}'),
                    subtitle: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Text('User: ${userData['name']}'),
                        Text('Email: ${userData['email']}'),
                        Text('Contact: ${userData['contactDetails']}'),
                        Text(
                            'Address: ${userData['houseNumber']},
${userData['streetName']}, ${userData['locality']}, ${userData['city']},
${userData['state']} - ${userData['pincode']}'),
                      ],
                    ),
                    trailing: IconButton(
                      icon: Icon(Icons.download),
                      onPressed: () {
                        final String? fileUrl = data['fileUrl'];
                        print('File URL: $fileUrl');
                        if (fileUrl != null && fileUrl is String) {
                          launch(fileUrl); // Open the URL in a new tab
                        } else {
                          ScaffoldMessenger.of(context).showSnackBar(
                            const SnackBar(content: Text('Invalid file
URL')),
                          );
                        }
                      },
                    ),
                  );
                },
              );
            }).toList(),
          );
        },
      ),
      bottomNavigationBar: BottomAppBar(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextButton(
              onPressed: () async {
                SharedPreferences prefs = await
SharedPreferences.getInstance();
```

```
                await FirebaseAuth.instance.signOut();
                prefs.remove('lastPage');
                Navigator.of(context).pushNamedAndRemoveUntil('/', (route)
=> false);
              },
              child: Text('Logout'),
            ),
          ],
        ),
      ),
    );
  }
}
```

**Profile_page.dart**

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:ex/launchpage.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class ProfilePage extends StatefulWidget {
  const ProfilePage({Key? key}) : super(key: key);

  @override
  _ProfilePageState createState() => _ProfilePageState();
}

class _ProfilePageState extends State<ProfilePage> {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  final TextEditingController nameController = TextEditingController();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController dobController = TextEditingController();
  final TextEditingController genderController = TextEditingController();
  final TextEditingController insuranceTypeController =
TextEditingController();
  final TextEditingController insuranceNumberController =
TextEditingController();
  final TextEditingController insuranceProviderController =
TextEditingController();

  final TextEditingController contactDetailsController =
TextEditingController();
  final TextEditingController houseNumberController =
TextEditingController();
  final TextEditingController streetNameController =
TextEditingController();
  final TextEditingController localityController =
TextEditingController();
  final TextEditingController cityController = TextEditingController();
  final TextEditingController stateController = TextEditingController();
  final TextEditingController pincodeController = TextEditingController();
  Future<void> _loadUserData() async {
    User? user = _auth.currentUser;
    if (user != null) {
      try {
        DocumentSnapshot userData =
            await _firestore.collection('users').doc(user.uid).get();
        if (userData.exists) {
          nameController.text = userData['name'] ?? '';
          emailController.text = userData['email'] ?? '';
          dobController.text = userData['dob'] ?? '';
          genderController.text = userData['gender'] ?? '';
          insuranceTypeController.text = userData['insuranceType'] ?? '';
          insuranceNumberController.text = userData['insuranceNumber'] ??
'';
          insuranceProviderController.text =
              userData['insuranceProvider'] ?? '';

          // Load communication details
          contactDetailsController.text = userData['contactDetails'] ??
'';
          houseNumberController.text = userData['houseNumber'] ?? '';
          streetNameController.text = userData['streetName'] ?? '';
          localityController.text = userData['locality'] ?? '';
          cityController.text = userData['city'] ?? '';
          stateController.text = userData['state'] ?? '';
          pincodeController.text = userData['pincode'] ?? '';
        }
      } catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Failed to load user data: $e')),
        );
      }
    }
  }

  Future<void> _updateUserData() async {
    User? user = _auth.currentUser;
    if (user != null) {
      try {
```

```
      await _firestore.collection('users').doc(user.uid).update({
        'name': nameController.text,
        'email': emailController.text,
        'dob': dobController.text,
        'gender': genderController.text,
        'insuranceType': insuranceTypeController.text,
        'insuranceNumber': insuranceNumberController.text,
        'insuranceProvider': insuranceProviderController.text,
        // Save communication details
        'contactDetails': contactDetailsController.text,
        'houseNumber': houseNumberController.text,
        'streetName': streetNameController.text,
        'locality': localityController.text,
        'city': cityController.text,
        'state': stateController.text,
        'pincode': pincodeController.text,
      });
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Profile updated successfully')),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to update profile: $e')),
      );
    }
  }
}

Future<void> _logout() async {
  await _auth.signOut();
  Navigator.of(context).pushAndRemoveUntil(
    MaterialPageRoute(builder: (context) => const LaunchPage()),
    (Route<dynamic> route) => false,
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
        title: const Text('Profile'),
        automaticallyImplyLeading: false,
        backgroundColor: Colors.teal,
      ),
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          FutureBuilder<void>(
            future: _loadUserData(),
            builder: (context, snapshot) {
              if (snapshot.connectionState == ConnectionState.waiting) {
                return const Center(child: CircularProgressIndicator());
              } else if (snapshot.hasError) {
                return Center(child: Text('Error: ${snapshot.error}'));
              } else {
                return Padding(
                  padding: const EdgeInsets.all(16.0),
                  child: ListView(
                    children: <Widget>[
                      const Text(
                        'Basic Details',
                        style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
                      ),
                      const SizedBox(height: 10),
                      TextField(
                        controller: nameController,
                        decoration: const InputDecoration(
                          labelText: 'Name',
                          border: OutlineInputBorder(),
                          filled: true,
                          fillColor: Colors.white,
                        ),
                      ),
                      const SizedBox(height: 10),
                      TextField(
                        controller: emailController,
                        decoration: const InputDecoration(
                          labelText: 'Email',
                          border: OutlineInputBorder(),
                          filled: true,
                          fillColor: Colors.white,
                        ),
                      ),
                      // Rest of your fields
                      const SizedBox(height: 10),
TextField(
controller: dobController,
decoration: const InputDecoration(
labelText: 'Date of Birth',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: genderController,
decoration: const InputDecoration(
labelText: 'Gender',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 20),
TextField(
controller: insuranceTypeController,
decoration: const InputDecoration(
labelText: 'Insurance Type',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: insuranceNumberController,
decoration: const InputDecoration(
labelText: 'Insurance Number',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: insuranceProviderController,
decoration: const InputDecoration(
labelText: 'Insurance Provider',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 20),
const Text(
'Communication Details',
style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
),
const SizedBox(height: 10),
TextField(
controller: contactDetailsController,
decoration: const InputDecoration(
labelText: 'Contact Details',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: houseNumberController,
decoration: const InputDecoration(
labelText: 'House Number',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: streetNameController,
decoration: const InputDecoration(
labelText: 'Street Name',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: localityController,
decoration: const InputDecoration(
labelText: 'Locality',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: cityController,
decoration: const InputDecoration(
labelText: 'City',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: stateController,
decoration: const InputDecoration(
labelText: 'State',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
TextField(
controller: pincodeController,
decoration: const InputDecoration(
labelText: 'Pincode',
border: OutlineInputBorder(),
filled: true,
fillColor: Colors.white,
),
),
const SizedBox(height: 10),
                      ElevatedButton(
                        onPressed: _updateUserData,
                        child: const Text('Update Profile'),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                      ),
                      const SizedBox(height: 20),
                      ElevatedButton(
                        onPressed: () {
                          Navigator.pop(context);
                        },
                        child: const Text('Back'),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                      ),
                      const SizedBox(height: 20),
                      ElevatedButton(
                        onPressed: _logout,
                        child: const Text('Logout'),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.teal,
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 50),
                          shadowColor: Colors.transparent,
                        ),
                      ),
                    ],
                  ),
                );
              }
            },
          ),
        ],
      ),
    );
  }
}
```

**Register_page.dart**

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:ex/launchpage.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

class RegisterPage extends StatefulWidget {
  const RegisterPage({Key? key}) : super(key: key);

  @override
  _RegisterPageState createState() => _RegisterPageState();
}

class _RegisterPageState extends State<RegisterPage> {
  final TextEditingController nameController = TextEditingController();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController =
TextEditingController();
  final TextEditingController dobController = TextEditingController();
  String gender = 'Male';
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: <Widget>[
          // Background Image
          Container(
            decoration: const BoxDecoration(
              image: DecorationImage(
                image: AssetImage("assets/launchpagebg.jpg"), // Correct
path to your image
                fit: BoxFit.cover,
              ),
            ),
          ),
          // Content
          Center(
            child: SingleChildScrollView(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  const Text(
                    'Hi There,',
                    style: TextStyle(fontSize: 36, fontWeight:
FontWeight.bold),
                  ),
                  const SizedBox(height: 8),
                  const Text(
                    'Enter your details to register your profile',
                    textAlign: TextAlign.center,
                    style: TextStyle(fontSize: 24),
                  ),
                  const SizedBox(height: 32),
                  TextField(
                    controller: nameController,
                    decoration: InputDecoration(
                      labelText: 'Name',
                      border: OutlineInputBorder(),
                      filled: true,
                      fillColor: Colors.white,
                    ),
                  ),
                  const SizedBox(height: 10),
                  TextField(
                    controller: emailController,
                    decoration: InputDecoration(
                      labelText: 'Email',
                      border: OutlineInputBorder(),
                      filled: true,
                      fillColor: Colors.white,
                    ),
                  ),
                  const SizedBox(height: 10),
                  TextField(
                    controller: passwordController,
                    decoration: InputDecoration(
                      labelText: 'Password',
                      border: OutlineInputBorder(),
                      filled: true,
                      fillColor: Colors.white,
                    ),
                    obscureText: true,
                  ),
                  const SizedBox(height: 10),
                  TextField(
                    controller: dobController,
                    decoration: InputDecoration(
                      labelText: 'Date of Birth',
                      border: OutlineInputBorder(),
                      filled: true,
                      fillColor: Colors.white,
                    ),
                    readOnly: true,
                    onTap: () async {
                      DateTime? pickedDate = await showDatePicker(
                        context: context,
                        initialDate: DateTime.now(),
                        firstDate: DateTime(1900),
                        lastDate: DateTime(2100),
                      );
                      if (pickedDate != null) {
                        String formattedDate =
DateFormat('yyyy-MM-dd').format(pickedDate);
                        setState(() {
                          dobController.text = formattedDate;
                        });
                      }
                    },
                  ),
                  const SizedBox(height: 10),
                  TextField(
                    controller: TextEditingController(text: gender),
                    readOnly: true,
                    decoration: InputDecoration(
                      labelText: 'Gender',
                      border: OutlineInputBorder(),
                      filled: true,
                      fillColor: Colors.white,
                      suffixIcon: Icon(Icons.arrow_drop_down),
                    ),
                    onTap: () {
                      showModalBottomSheet(
                        context: context,
                        builder: (BuildContext context) {
                          return Container(
                            height: MediaQuery.of(context).size.height *
0.3,

                            child: Column(
                              children: [
                                ListTile(
                                  title: Text('Male'),
                                  onTap: () {
                                    setState(() {
                                      gender = 'Male';
                                    });
                                    Navigator.pop(context);
                                  },
                                ),
                                ListTile(
                                  title: Text('Female'),
                                  onTap: () {
                                    setState(() {
                                      gender = 'Female';
                                    });
                                    Navigator.pop(context);
                                  },
                                ),
                                ListTile(
                                  title: Text('Other'),
                                  onTap: () {
                                    setState(() {
                                      gender = 'Other';
                                    });
                                    Navigator.pop(context);
                                  },
                                ),
                              ],
                            ),
                          );
                        },
                      );
                    },
                  ),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: () async {
                      try {
                        print('Starting registration process...');
                        UserCredential userCredential = await
_auth.createUserWithEmailAndPassword(
                          email: emailController.text,
                          password: passwordController.text,
                        );
                        User? user = userCredential.user;
                        if (user != null) {
                          print('User created with UID: ${user.uid}');
                          await
_firestore.collection('users').doc(user.uid).set({
                            'name': nameController.text,
                            'email': emailController.text,
                            'dob': dobController.text,
                            'gender': gender,
                          });
                          print('User data saved to Firestore');
                          ScaffoldMessenger.of(context).showSnackBar(
                            const SnackBar(content: Text('Registration
successful')),
                          );
                          print('Navigating to LaunchPage...');
                          Navigator.pushReplacement(
                            context,
                            MaterialPageRoute(builder: (context) => const
LaunchPage()),
                          );
                        }
                      } catch (e) {
                        print('Registration failed: $e');
                        ScaffoldMessenger.of(context).showSnackBar(
                          SnackBar(content: Text('Registration failed:
$e')),
                        );
                      }
                    },
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal, // Changed to teal
                      foregroundColor: Colors.white, // Text color
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30), //
Capsule shape
                      ),
                    ),
                    child: const Text('Register', style:
TextStyle(fontSize: 16)),
                  ),
                  const SizedBox(height: 10),
                  ElevatedButton(
                    onPressed: () {
                      Navigator.pop(context); // Go back to the previous
page
                    },
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.teal, // Changed to teal
                      foregroundColor: Colors.white, // Text color
                      padding: const EdgeInsets.symmetric(horizontal: 40,
vertical: 15),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(30), //
Capsule shape
                      ),
                    ),
                    child: const Text('Back', style: TextStyle(fontSize:
16)),
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

**Request_medical _details.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';

class RequestMedicalDetailsPage extends StatefulWidget {
  const RequestMedicalDetailsPage({Key? key}) : super(key: key);

  @override
  _RequestMedicalDetailsPageState createState() =>
      _RequestMedicalDetailsPageState();
}

class _RequestMedicalDetailsPageState
    extends State<RequestMedicalDetailsPage> {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final FirebaseStorage _storage = FirebaseStorage.instance;
  Map<String, bool> selectedFiles = {};
  TextEditingController _messageController = TextEditingController();

  Future<List<Map<String, dynamic>>> _fetchUserFiles() async {
    try {
      List<Map<String, dynamic>> userFiles = [];
      QuerySnapshot usersSnapshot = await
_firestore.collection('users').get();
      for (var userDoc in usersSnapshot.docs) {
        String userId = userDoc.id;
        String userName = userDoc['name'];
        String userEmail = userDoc['email'];
        try {
          ListResult filesResult =
              await _storage.ref('uploads/$userId').listAll();
          for (var fileRef in filesResult.items) {
            userFiles.add({
              'userId': userId,
              'userName': userName,
              'userEmail': userEmail,
              'fileName': fileRef.name,
              'filePath': fileRef.fullPath,
            });
          }
        } catch (e) {
          debugPrint('Error fetching files for user $userId: $e');
        }
      }
      return userFiles;
    } catch (e) {
      debugPrint('Error fetching user files: $e');
      throw e;
    }
  }

  void _sendRequest(String message) async {
    try {
      List<Map<String, dynamic>> requestedFiles = selectedFiles.entries
          .where((entry) => entry.value)
          .map((entry) => {
                'filePath': entry.key,
              })
          .toList();

      for (var request in requestedFiles) {
        // Extract userId from filePath
        List<String> pathComponents = request['filePath'].split('/');
        String userId = pathComponents[1];

        // Add request to requests collection
        await _firestore.collection('requests').add({
          'userId': userId,
          'filePath': request['filePath'],
          'message': message,
          'status': 'pending',
          'timestamp': FieldValue.serverTimestamp(),
        });

        // Send message to user
        await _firestore.collection('messages').add({
          'userId': userId,
          'message': message,
          'timestamp': FieldValue.serverTimestamp(),
        });
      }
      // Show a confirmation message
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Request sent successfully')),
      );
      Navigator.of(context).pop();
    } catch (e) {
      debugPrint('Error sending request: $e');
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text('Failed to send request')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Request Medical Details')),
      body: Column(
        children: [
          Expanded(
            child: FutureBuilder<List<Map<String, dynamic>>>(
              future: _fetchUserFiles(),
              builder: (context, snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                }
                if (snapshot.hasError) {
                  return Center(child: Text('Error: ${snapshot.error}'));
                }
                if (!snapshot.hasData || snapshot.data!.isEmpty) {
                  return const Center(child: Text('No files found'));
                }
                return ListView(
                  children: snapshot.data!.map((file) {
                    return CheckboxListTile(
                      title: Text(
                          '${file['userName']} (${file['userEmail']}) -
${file['fileName']}'),
                      value: selectedFiles[file['filePath']] ?? false,
                      onChanged: (value) {
                        setState(() {
                          selectedFiles[file['filePath']] = value!;
                        });
                      },
                    );
                  }).toList(),
                );
              },
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: TextField(
              controller: _messageController,
              decoration: InputDecoration(
                hintText: 'Enter a message',
                border: OutlineInputBorder(),
              ),
            ),
          ),
          ElevatedButton(
            onPressed: () {
              if (selectedFiles.isEmpty) {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                      content: Text('Please select at least one file')),
                );
              } else {
                _sendRequest(_messageController.text);
              }
            },
            child: const Text('Send Request'),
          ),
        ],
      ),
    );
  }
}
```

**Send_approved_requests.dart**

```
import 'package:flutter/material.dart';
import 'package:file_picker/file_picker.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:firebase_auth/firebase_auth.dart';

class SendApprovedReportsPage extends StatefulWidget {
  const SendApprovedReportsPage({super.key});

  @override
  _SendApprovedReportsPageState createState() =>
_SendApprovedReportsPageState();
}

class _SendApprovedReportsPageState extends State<SendApprovedReportsPage>
{
  bool _isOnline = true;
  String? _userId;
  PlatformFile? _pickedFile;

  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  final FirebaseStorage _storage = FirebaseStorage.instance;
  final FirebaseAuth _auth = FirebaseAuth.instance;

  Future<void> _pickFile() async {
    FilePickerResult? result = await FilePicker.platform.pickFiles();

    if (result != null) {
      setState(() {
        _pickedFile = result.files.first;
      });
    }
  }

  Future<void> _sendReport() async {
    if (_isOnline && _userId != null && _pickedFile != null) {
      // Upload the file to Firebase Storage
      try {
        final fileRef =
_storage.ref().child('reports/${_pickedFile!.name}');
        await fileRef.putData(_pickedFile!.bytes!);

        final fileUrl = await fileRef.getDownloadURL();

        // Update Firestore with the report information
        await _firestore.collection('reports').add({
          'userId': _userId,
          'fileUrl': fileUrl,
          'uploadedAt': Timestamp.now(),
          'uploadedBy': _auth.currentUser?.uid,
        });

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Report sent to user $_userId')),
        );
      } catch (e) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Failed to send report: $e')),
        );
      }
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Please fill in all fields and pick a
file')),
      );
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Send Approved Reports'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
```

```dart
Row(
  children: [
    Expanded(
      child: ListTile(
        title: const Text('Send Online'),
        leading: Radio<bool>(
          value: true,
          groupValue: _isOnline,
          onChanged: (bool? value) {
            setState(() {
              _isOnline = value!;
            });
          },
        ),
      ),
    ),
    Expanded(
      child: ListTile(
        title: const Text('Send Offline'),
        leading: Radio<bool>(
          value: false,
          groupValue: _isOnline,
          onChanged: (bool? value) {
            setState(() {
              _isOnline = value!;
            });
          },
        ),
      ),
    ),
  ],
),
if (_isOnline)
  TextField(
    decoration: const InputDecoration(labelText: 'User ID'),
    onChanged: (value) {
      setState(() {
        _userId = value;
      });
    },
              ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: _pickFile,
              child: const Text('Pick File'),
            ),
            if (_pickedFile != null)
              Padding(
                padding: const EdgeInsets.symmetric(vertical: 8.0),
                child: Text('Picked file: ${_pickedFile!.name}'),
              ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: _sendReport,
              child: const Text('Send Report'),
            ),
          ],
        ),
      ),
    );
  }
}
```

**View_appointment.dart**

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class ViewAppointmentsPage extends StatelessWidget {
  const ViewAppointmentsPage({Key? key});

  @override
  Widget build(BuildContext context) {
    final FirebaseAuth _auth = FirebaseAuth.instance;
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;

    User? user = _auth.currentUser;
    if (user == null) {
      return Scaffold(
        appBar: AppBar(
          title: const Text('View Appointments'),
          backgroundColor: Colors.teal,
        ),
        body: const Center(child: Text('Please log in to view your
appointments')),
      );
    }
return Scaffold(
  appBar: AppBar(
    title: const Text('View Appointments'),
    backgroundColor: Colors.teal,
  ),
  body: Stack(
    children: <Widget>[
      // Background Image
      Container(
        decoration: const BoxDecoration(
          image: DecorationImage(
            image: AssetImage("assets/launchpagebg.jpg"),
            fit: BoxFit.cover,
          ),
        ),
      ),
      // Content
      Container(
        margin: const EdgeInsets.all(16.0),
        child: StreamBuilder<QuerySnapshot>(
          stream: _firestore.collection('appointments').where('userId',
isEqualTo: user.uid).snapshots(),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return const Center(child: CircularProgressIndicator());
            }
            if (snapshot.hasError) {
              return Center(child: Text('Error: ${snapshot.error}'));
            }
            if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
              return const Center(child: Text('No appointments found'));
            }
            return ListView(
              children: snapshot.data!.docs.map((doc) {
                var data = doc.data() as Map<String, dynamic>;
                return Card(
                  margin: const EdgeInsets.symmetric(horizontal: 16.0,
vertical: 8.0),
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10.0),
                    side: BorderSide(color: Colors.white),
                  ),
                  color: Colors.white,
                  child: ListTile(
                    title: Text(
                      '${data['date']} - ${data['time']}',
                      style: TextStyle(color: Colors.black),
                    ),
                    subtitle: Text(
                      '${data['specialization']} - Dr.
${data['doctorName']}',
                      style: TextStyle(color: Colors.black),
                    ),
                  ),
                );
              }).toList(),
            );
          },
        ),
      ),
    ],
  ),
);
  }
}
```

**View_submitted_reports.dart**

```
import 'package:flutter/material.dart';

class ViewSubmittedReportsPage extends StatelessWidget {
  const ViewSubmittedReportsPage({super.key});

  @override
  Widget build(BuildContext context) {
    // Add your code to view submitted reports
    return Scaffold(
      appBar: AppBar(
        title: const Text('Submitted Reports'),
      ),
      body: Center(
        child: const Text('Here you can view submitted reports.'),
      ),
    );
  }
}
```

# Chapter Five : Testing

Rigorous testing methodologies are employed to ensure the reliability, functionality, and performance of theMedisync application across various scenarios and use cases.

**1. Unit Testing:**

- Focuses on testing individual components (units) of the application in isolation.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:medisync/widgets/some_widget.dart'; // replace with actual
widget
import 'package:medisync/utils/some_util.dart'; // replace with actual
utility

void main() {
  group('Unit Tests', () {
    test('Utility function test', () {
      final result = someUtilityFunction();
      expect(result, expectedValue);
    });

    testWidgets('Some Widget test', (WidgetTester tester) async {
      await tester.pumpWidget(SomeWidget());

      expect(find.text('Expected Text'), findsOneWidget);
    });
  });
}
```

- Verifies that each unit behaves as expected and meets its design specifications.

**2. Integration Testing:**

- Tests the interaction between integrated components or modules of the application.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:medisync/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Integration Tests', () {
    testWidgets('Full app test', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();

      expect(find.text('Welcome to MediSync!'), findsOneWidget);

      await tester.tap(find.text('Book Doctor Appointment'));
      await tester.pumpAndSettle();
      expect(find.text('Doctor Appointment Page'), findsOneWidget);
    });
  });
}
```

- Ensures that different parts of the system work together seamlessly as a whole.

**3. Smoke Testing:**

- Conducted to verify that essential functionalities of the application work correctly without detailed testing.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:medisync/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Intensive Smoke Tests for MediSync', () {
    testWidgets('App loads correctly and displays the homepage',
(WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      expect(find.text('Welcome to MediSync!'), findsOneWidget);
    });

    testWidgets('Book Doctor Appointment Page loads', (WidgetTester
tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('Book Doctor Appointment'));
      await tester.pumpAndSettle();
      expect(find.text('Doctor Appointment Page'), findsOneWidget);
    });

    testWidgets('Book Lab Test Page loads', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('Book Lab Test'));
      await tester.pumpAndSettle();
      expect(find.text('Lab Test Page'), findsOneWidget);
    });

    testWidgets('Upload Health Records functionality', (WidgetTester
tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('Upload Health Records'));
      await tester.pumpAndSettle();
      // Assume file picker and upload process is mocked
      expect(find.text('Uploading...'), findsOneWidget);
    });

    testWidgets('Emergency Care request', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('EMERGENCY CARE'));
      await tester.pumpAndSettle();
      expect(find.text('Request Received'), findsOneWidget);
    });

    testWidgets('ChatBot interaction', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.byIcon(Icons.chat));
      await tester.pumpAndSettle();
      expect(find.text('ChatBot'), findsOneWidget);
    });
  });
}
```

- Ensures basic functionalities like login, navigation, and main feature accessibility are operational.

**4. Regression Testing:**

- Re-runs previously conducted tests to ensure that recent code changes have not adversely affected existing functionalities.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:medisync/main.dart' as app;
```

```
void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Regression Tests', () {
    testWidgets('App loads correctly', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      expect(find.text('Welcome to MediSync!'), findsOneWidget);
    });

    // Add more tests for previously tested functionalities
  });
}
```

- Validates that new updates or fixes do not introduce unintended side effects or issues.

**5. Black Box Testing:**

- Tests the application's functionality without detailed knowledge of its internal code structure or implementation.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:medisync/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Black-Box Tests', () {
    testWidgets('User can book a doctor appointment', (WidgetTester
tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('Book Doctor Appointment'));
      await tester.pumpAndSettle();
      expect(find.text('Doctor Appointment Page'), findsOneWidget);
    });

    testWidgets('User can order medicines', (WidgetTester tester) async {
      app.main();
      await tester.pumpAndSettle();
      await tester.tap(find.text('Order Medicines'));
      await tester.pumpAndSettle();
      expect(find.text('Order Medicines Page'), findsOneWidget);
    });
  });
}
```
- Focuses on validating outputs based on inputs and system behavior as perceived by users.

**6. White Box Testing:**
- Examines the internal structure and workings of the application's code.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:medisync/utils/some_util.dart';

void main() {
  group('White-Box Tests', () {
    test('Test some utility function logic', () {
      final result = someUtilityFunction();
      expect(result, expectedValue);
    });

    test('Test another internal function', () {
      final result = anotherInternalFunction();
      expect(result, anotherExpectedValue);
    });
  });
}
```

- Verifies the logic, paths, and interactions within the codebase to ensure completeness and correctness.

**7. Performance Testing:**
- Evaluates how the application performs under various conditions, such as load, stress, and responsiveness.

```
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:medisync/main.dart' as app';

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group('Performance Tests', () {
    testWidgets('Performance test for loading home page', (WidgetTester
tester) async {
      app.main();

      final stopwatch = Stopwatch()..start();
      await tester.pumpAndSettle();
      stopwatch.stop();

      print('Time taken to load home page:
${stopwatch.elapsedMilliseconds}ms');
      expect(stopwatch.elapsedMilliseconds, lessThan(2000));
    });
  });
}
```

- Measures response times, resource usage, and scalability to ensure optimal performance for users.

Each testing method plays a crucial role in ensuring thatMedisync meets high standards of functionality, reliability, and performance. Through comprehensive testing, potential issues are identified and addressed early in the development lifecycle, ensuring a robust and user-friendly healthcare management system.

## Future Scope

The potential forMedisync to evolve and expand is vast, given the rapidly advancing landscape of healthcare technology and patient care. Here are several key areas for future development:

**1. Enhanced AI and Machine Learning Integration:**

- Implement advanced AI algorithms for predictive analytics, helping to identify potential health issues before they become critical.

- Develop personalized health recommendations and treatment plans based on patient data and history.

**2. Telemedicine and Remote Monitoring:**
- Integrate telemedicine capabilities to allow virtual consultations with healthcare providers.
- Develop remote patient monitoring features using IoT devices to track vital signs and other health metrics in real time.

**3. Expanded Insurance and Billing Integration:**
- Enhance the insurance module to include direct billing capabilities, allowing for seamless processing of insurance claims and payments.
- Integrate with a wider range of insurance providers and health plans for broader coverage.

**4. Advanced Data Analytics and Reporting:**
- Implement comprehensive data analytics tools to generate detailed health reports and insights for users.
- Develop reporting features for healthcare providers to analyze patient trends and outcomes.

**5. Multi-Language Support:**
- Expand the application's accessibility by adding multi-language support to cater to a global user base.
- Provide localization features to adapt the application to different cultural and regional requirements.

**6. Enhanced Security Measures:**
- Continuously update and enhance security protocols to stay ahead of emerging threats and ensure the highest level of data protection.
- Implement biometric authentication methods, such as fingerprint or facial recognition, for added security.

**7. Interoperability with Other Healthcare Systems:**
- Develop APIs and integration capabilities to ensure seamless interoperability with other electronic health record (EHR) systems and healthcare applications.
- Facilitate data exchange and collaboration among different healthcare providers and institutions.

**8. User Experience Enhancements:**

- Continuously improve the user interface and user experience based on user feedback and evolving design trends.
- Implement features such as voice commands and accessibility options for users with disabilities.

**9. Community and Support Features:**
- Develop community features where users can connect with others who have similar health conditions or concerns.
- Provide support resources, such as access to medical articles, videos, and forums moderated by healthcare professionals.

**10. Regulatory Compliance and Certifications:**
- Ensure ongoing compliance with healthcare regulations and standards such as HIPAA, GDPR, and others.
- Pursue certifications and endorsements from healthcare authorities and organizations to build trust and credibility.

By pursuing these future developments,Medisync can continue to innovate and adapt to the ever-changing healthcare environment, providing users with an even more powerful, versatile, and secure platform for managing their health and well-being.

# Conclusion

The development ofMedisync has been a comprehensive journey, aiming to revolutionize the way healthcare services are managed and delivered. By integrating modern technologies such as Firebase Firestore, Flutter SDK, and advanced security measures like AES 256 encryption, the application provides a secure, user-friendly platform for patients, healthcare providers, and pharmacists.

Throughout the project's lifecycle, from initial planning and rigorous testing to detailed modeling and implementation, every phase has been meticulously executed to ensure the application meets high standards of reliability, functionality, and performance. The use of various testing methodologies, including unit, integration, smoke, regression, black box, white box, and performance testing, has ensured that the application is robust and ready for real-world use.

The modular design ofMedisync allows for scalability and future enhancements, making it adaptable to evolving healthcare needs. Key features such as secure medical records access, an emergency button, appointment scheduling, insurance management, and a pharmacy module contribute to a comprehensive healthcare management system that prioritizes user convenience and data security.

As we conclude this project,Medisync stands as a testament to the effective use of modern technologies and collaborative efforts in addressing complex healthcare challenges. The application not only streamlines healthcare management but also empowers users with greater control over their health information and services. Looking ahead,Medisync is poised to make a significant impact in the healthcare industry, providing a unified, efficient, and secure platform for all stakeholders involved.

## References

**1. Electronic Health Records (EHR) Integration:**
- Buntin, M. B., Burke, M. F., Hoaglin, M. C., & Blumenthal, D. (2011). The benefits of health information technology: a review of the recent literature shows predominantly positive results. *Health Affairs, 30*(3), 464-471. (https://www.healthaffairs.org/doi/full/10.1377/hlthaff.2011.0178)

**2. Secure Data Transmission in Healthcare:**
- Thong, D. D., Tran, M. T., & Nguyen, N. T. (2018). A security model for health information systems based on cloud computing. *International Journal of Advanced Computer Science and Applications (IJACSA), 9*(10).
(https://thesai.org/Publications/ViewPaper?Volume=9&Issue=10&Code=IJACSA&SerialNo=63)

**3. Cloud Computing in Healthcare:**
- Rolim, C. O., Koch, F. L., Westphall, C. B., Werner, J., & Fracalossi, A. (2010). A cloud computing solution for patient's data collection in health care institutions. *2010 Second International Conference on eHealth, Telemedicine, and Social Medicine*, 95-99.
(https://ieeexplore.ieee.org/document/5431663)

**4. Telemedicine Implementation:**
- Dorsey, E. R., & Topol, E. J. (2020). Telemedicine 2020 and the next decade. *The Lancet,395*(10227),859.
(https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)30424-4/fulltext)

**5. IoT in Healthcare:**

- Islam, S. M. R., Kwak, D., Kabir, M. H., Hossain, M., & Kwak, K. S. (2015). The Internet of Things for health care: a comprehensive survey. *IEEE Access, 3*, 678-708. (https://ieeexplore.ieee.org/document/7113786)

**6. AI and Machine Learning in Healthcare:**
- Obermeyer, Z., & Emanuel, E. J. (2016). Predicting the future—big data, machine learning, and clinical medicine. *The New England Journal of Medicine, 375*(13), 1216-1219. (https://www.nejm.org/doi/full/10.1056/NEJMp1606181)

**7. Patient-Centric Healthcare Systems:**
- Eysenbach, G. (2001). What is e-health? *Journal of Medical Internet Research, 3*(2), e20. (https://www.jmir.org/2001/2/e20/)

**8. Data Privacy in Healthcare:**
- Rindfleisch, T. C. (1997). Privacy, information technology, and health care. *Communications of the ACM, 40*(8), 92-100. (https://dl.acm.org/doi/10.1145/257874.257896)

**9. Mobile Health Applications:**
- Kumar, S., & Nilsen, W. J. (2016). Mobile health technology evaluation: the mHealth evidence workshop. *American Journal of Preventive Medicine, 45*(2), 228-236. (https://www.ajpmonline.org/article/S0749-3797(13)00356-2/fulltext)

**10. Blockchain for Health Data and Its Potential Use:**
- Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). MedRec: Using Blockchain for Medical Data Access and Permission Management. *2016 2nd International Conference on Open and Big Data (OBD)*, 25-30. (https://ieeexplore.ieee.org/document/7573685)