Technological Institute of the Philippines - Quezon City

College of Computer Studies

Bachelor of Science in Computer Science

SafeSpace

In Partial Fulfillment

of the Requirement for the Course

ITE 010A - Human Computer and Interaction

Presented by:

Limson, CLark Czedrick F.

Lituania, Andrea A.

Marasigan, Ma. Clarissa C.

Section:

CS32S2

Presented to:

Mr. Jess Garcia

May 2024

TABLE OF CONTENTS

I. Introduction

- **Background of the Study**
- **Objectives**
- **Significance of the Study**
- **Scope and Delimitation**

II. Hardware and Software Specification

- **Hardware**
- **Software**
- **Screenshots with Description**
- **Source Code**

III. Group Member's Summary of Assigned Task

I. Introduction

Background of the Study

Youths are facing numerous challenges ranging from academic pressure to social stigma, and especially, mental health issues. In today's fast-paced and technology-driven society, young people are increasingly being dependent on mobile applications for support and guidance. However, many existing systems lack the necessary features to effectively address the unique needs of this demographic.

Recognizing the need for a safe and accessible platform for youths to address these concerns, our study focuses on understanding the problems encountered by youth in accessing support and guidance through technology. These problems may include issues such as lack of confidentiality, fear of judgment, limited access to resources, and difficulty in finding relevant and trustworthy information. By identifying and addressing these challenges, our goal is to develop a user-centered mobile application. This mobile application aims to provide a confidential and supportive environment for youth to express their feelings, seek advice, and connect with trained professionals. Through innovative technology and user-centered design principles, the application seeks to overcome the limitation of existing systems and provide a safe and accessible platform for youth to address their mental and emotional well-being.

Project Objectives

The objectives of the SafeSpace Mobile Application are:

1. To provide a secure and confidential platform. Ensuring that young people feel safe to express their thoughts and feelings without fear of judgment or privacy breaches.
2. To facilitate open and empathetic dialogue among users through anonymous group chat and forums, encouraging peer support and connection.
3. To offer easily accessible tools, resources, and professional support to help users manage their mental and emotional well-being effectively.
4. To educate users and promote understanding and acceptance of mental health issues, creating a culture of support and empathy within the community.
5. To empower users to address their challenges proactively, develop coping strategies, and foster resilience in navigating life's ups and downs.

Significance of the Study

This study is about a mobile application for youth that aims to address their mental and emotional well-being. The proponents named it 'SafeSpace' because it provides a supportive community and confidentiality within the mobile application. The proponents believe that the culmination of this study will be beneficial to the following:

Youth. This mobile application will provide the youth with access to a safe space where they can express themselves, seek support, and learn coping strategies for emotional and mental well-being. This can contribute to their overall mental health and resilience.

Parents. This mobile application will benefit parents by providing them a tool to support their children's emotional and mental health. By offering tools, advice, and insights into their children's wellbeing, the aforementioned mobile application will assist parents in better understanding and meeting their children's needs.

Educators. This mobile application will be beneficial to the educators as a resource that can complement their efforts in promoting mental health awareness and support among students. By providing tools for recognizing and resolving emotional difficulties, the aforementioned mobile application helps to improve the learning environment.

Mental Health Professionals. This mobile application will aid professionals such as psychiatrists and therapists by extending their reach beyond traditional therapy settings. They can also recommend the mobile application to clients as a supplementary resource for ongoing support and skill-building.

Policymakers. This mobile application will be beneficial to the policymakers by addressing the youth mental health that can lead to healthier and more resilient communities.

Proponents. This study will help the proponents enhance their future professional growth giving them career opportunities that align with the field of the study.

Future Researchers. This study will help the future researchers as their reference for new research related to a certain topic and introduce innovative ideas and research methods.

Scope and Limitation

In a research project, it is essential to identify the scope and limitations of the study in order to provide understanding on what topics will be included and which will not. This section provides a guide for both researchers and readers, providing reasonable expectations and limits for the research project.

Scope

The proponents developed a mental health mobile application which prioritizes creating a safe and accessible space for users to manage their mental well-being. The following are the core features which empower the users. Built-in calendar for tracking moods, schedule appointments, and set reminders for self-care activities. There's also a real-time chat with Counselors and other Users while maintaining anonymity. The users can also track their emotional state over time to identify patterns and triggers. Furthermore, when a user clicks on a certain mood, the application will direct them to available counselors for a session, facilitating timely and appropriate support. Additionally, the application provides users with real-time access to the availability of counselors which enables them to schedule sessions conveniently based on the counselors' current schedules. The users can also express themselves through journaling where they can record their thoughts and experience and monitor progress through journaling entries.

Limitation

However, to ensure user privacy and focus on core functionality, some features are currently limited. The application is currently only accessible through the mobile app, with no web browser version available. Articles and other educational materials to support mental health are not yet included. To maintain a safe and manageable environment, users can only participate in the available group chats, with no option to create additional private groups. The app also currently lacks notification functionality to alert the users of messages or updates. While the emergency hotline is displayed, the application does not offer direct, in-app connection to these resources. Additional password security measures beyond basic login credentials are not yet implemented. The video and call functionalities for direct communication with

counselors are not available. Lastly, the users were not able to compensate counselors on a per-session basis through the application.

II. Hardware and Software Specifications

Hardware	Software
Processor: Quad-core 1.4 GHz or higher **RAM:** Minimum of 2GB Ram **Storage:** At least 50 MB free space for installation **Screen Resolution:** 1280x720 pixel or higher **Compatibility:** **Device:** Android Smartphone or Tablet **Orientation:** Support portrait orientation	**Android version:** Android 10 up to latest Android version **Permissions:** Read/Write Internal Storage

Screenshots with Descriptions

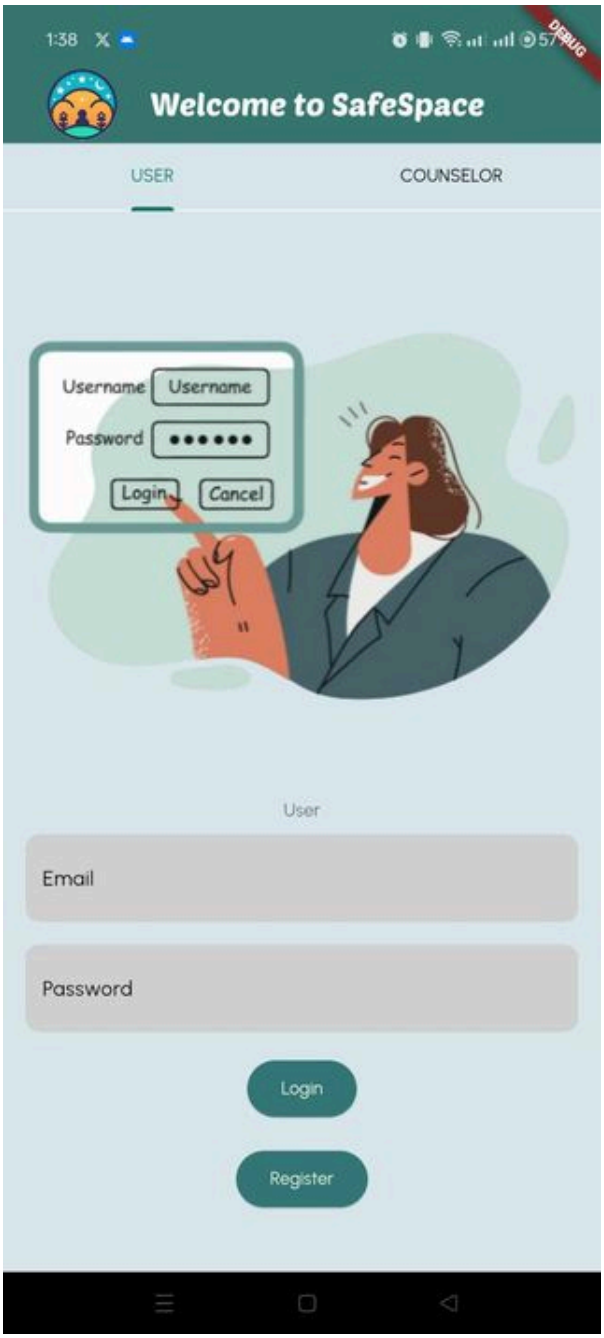


Figure 1 | Login

In this screen, the user/patient may login to their account or register if they don't have an account yet or are new to the app.

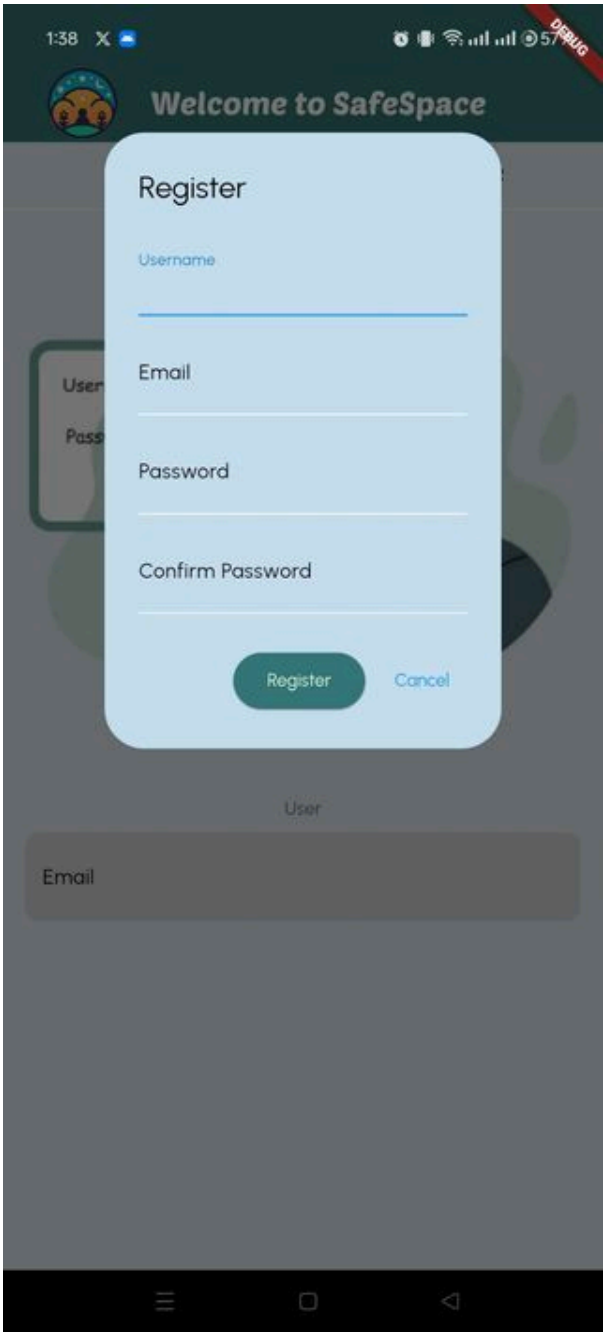


Figure 2 | Register

In this screen, if the user/patient chooses the register in Figure 1, they will have to fill up the Username, Email, Password, and Confirm Password for them to create an account.

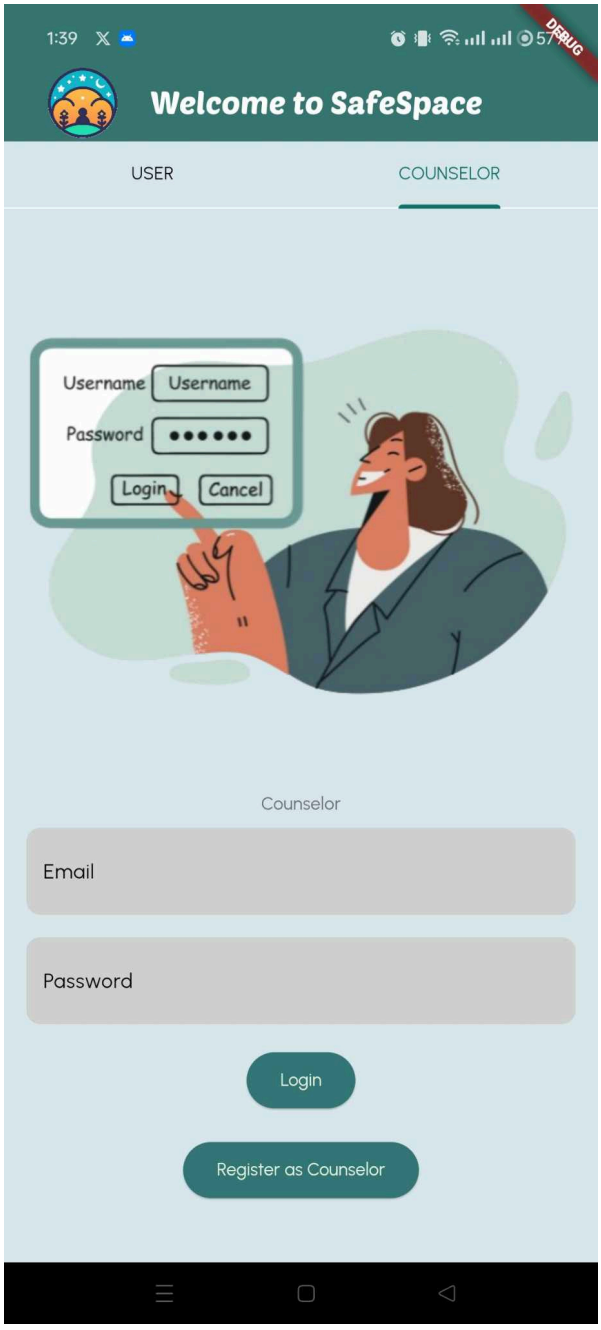


Figure 3 | Counselor Login

This is the login screen for the counselor of the app. In this screen, they can choose to login if they already have an account or register as a counselor is they don't have an account.

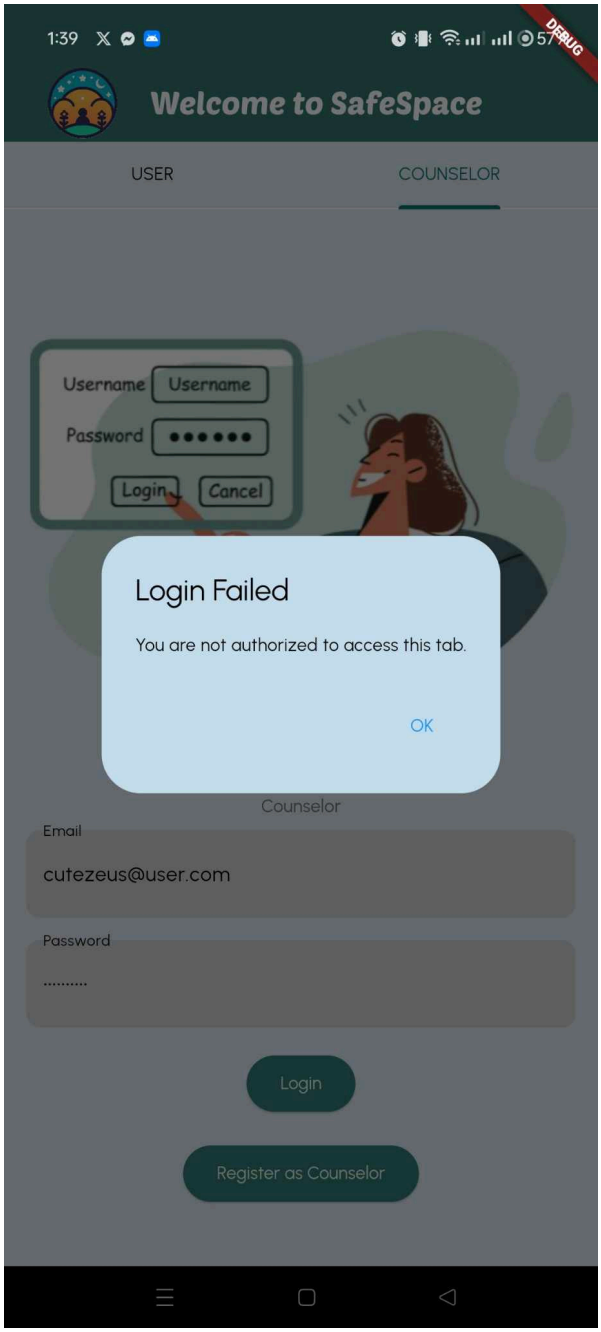
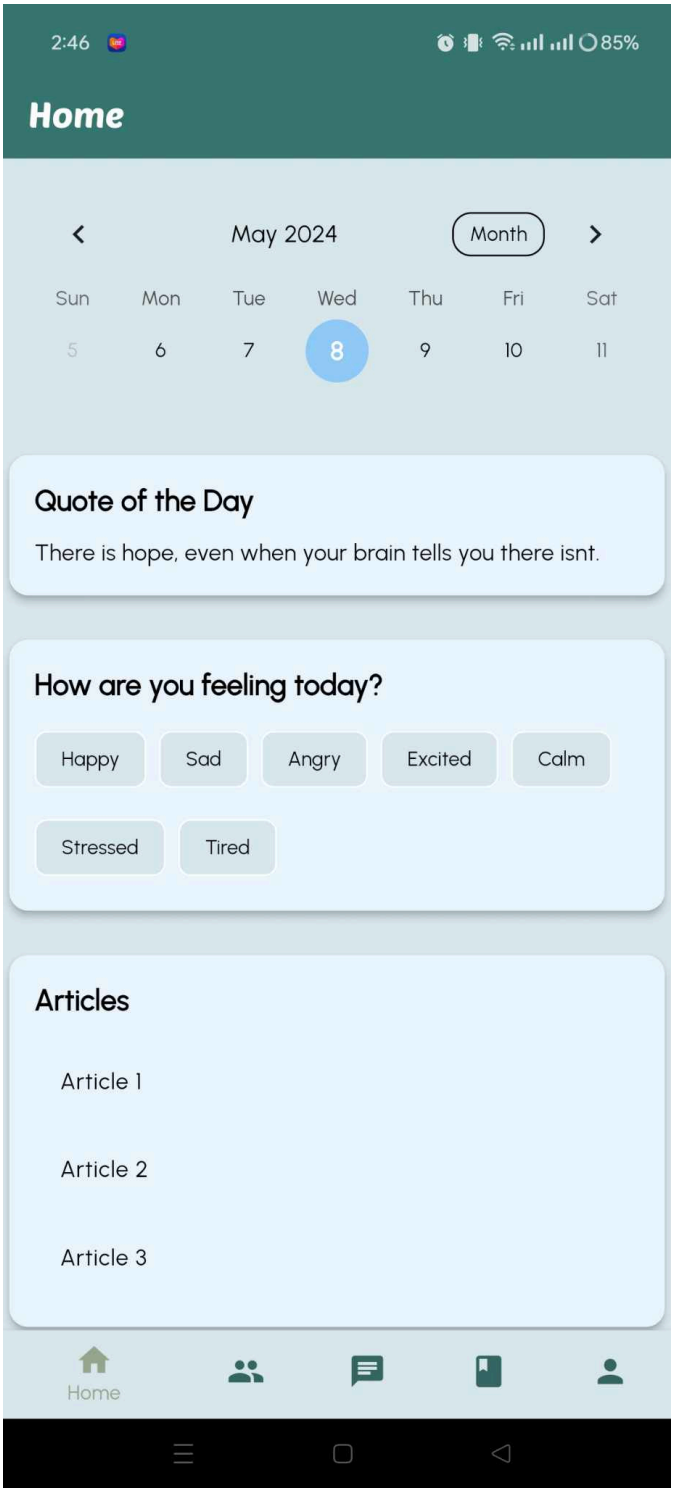


Figure 4 | Login Failed Prompt

This prompt will appear whenever an unauthorized user tries to login as a counselor. This may also appear if a user/patient tries to login as a counselor when their email is already registered as an ordinary user.



Figuro 5: Home Page (User)

This is the home page of a user/patient where it includes the built-in calendar, quote of the day, mood track and articles related to mental health.

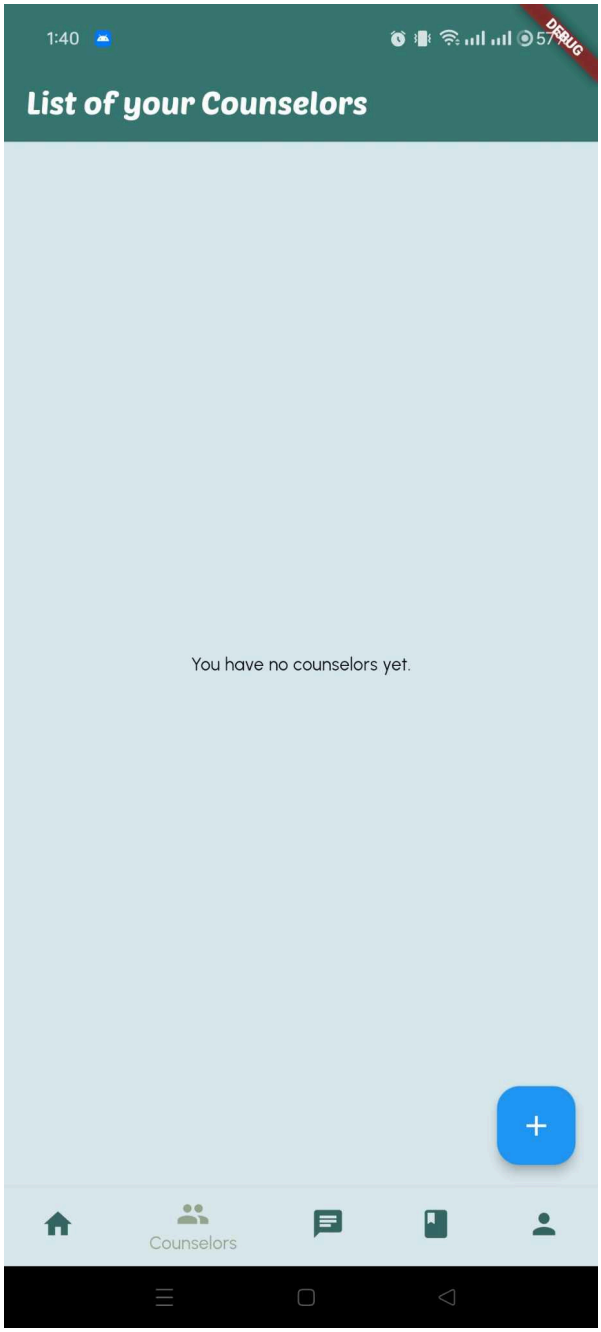


Figure 5 | Counselor Page

In this screen, the user/patients will see the list of the counselors that they have chosen.

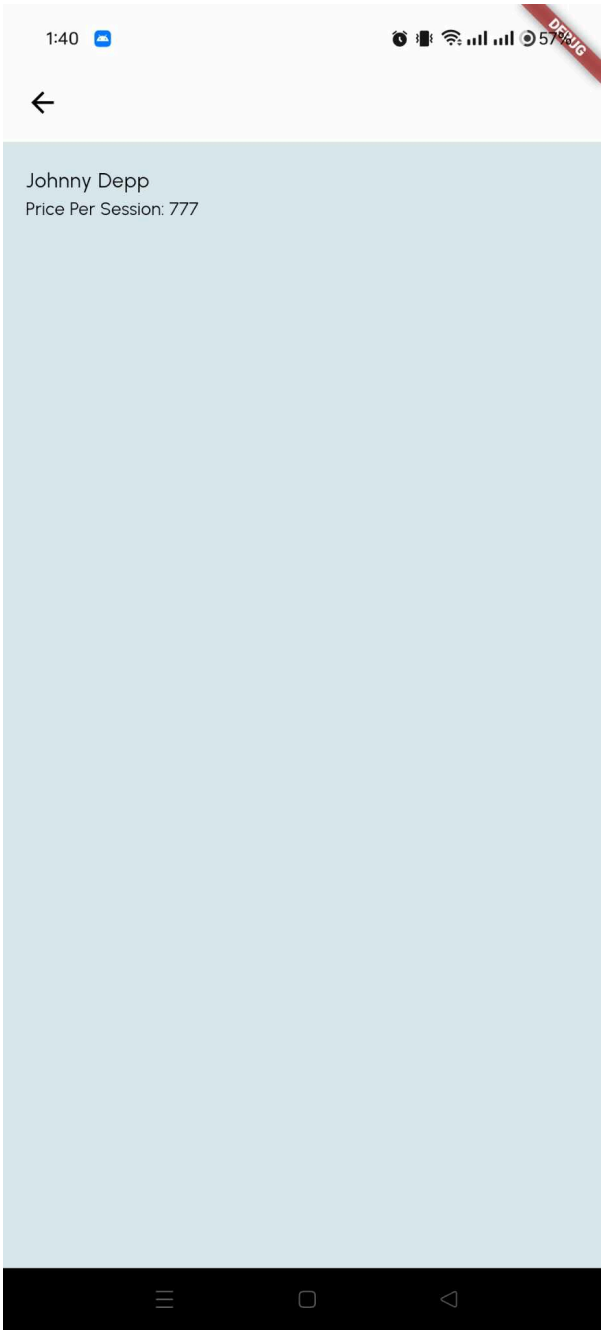


Figure 6 | List of Counselors

If the user/patient clicks on the 'plus' button in Figure 5, this is the screen that will appear. Here, they will see the list of counselors that they can choose to talk to and add to your list.

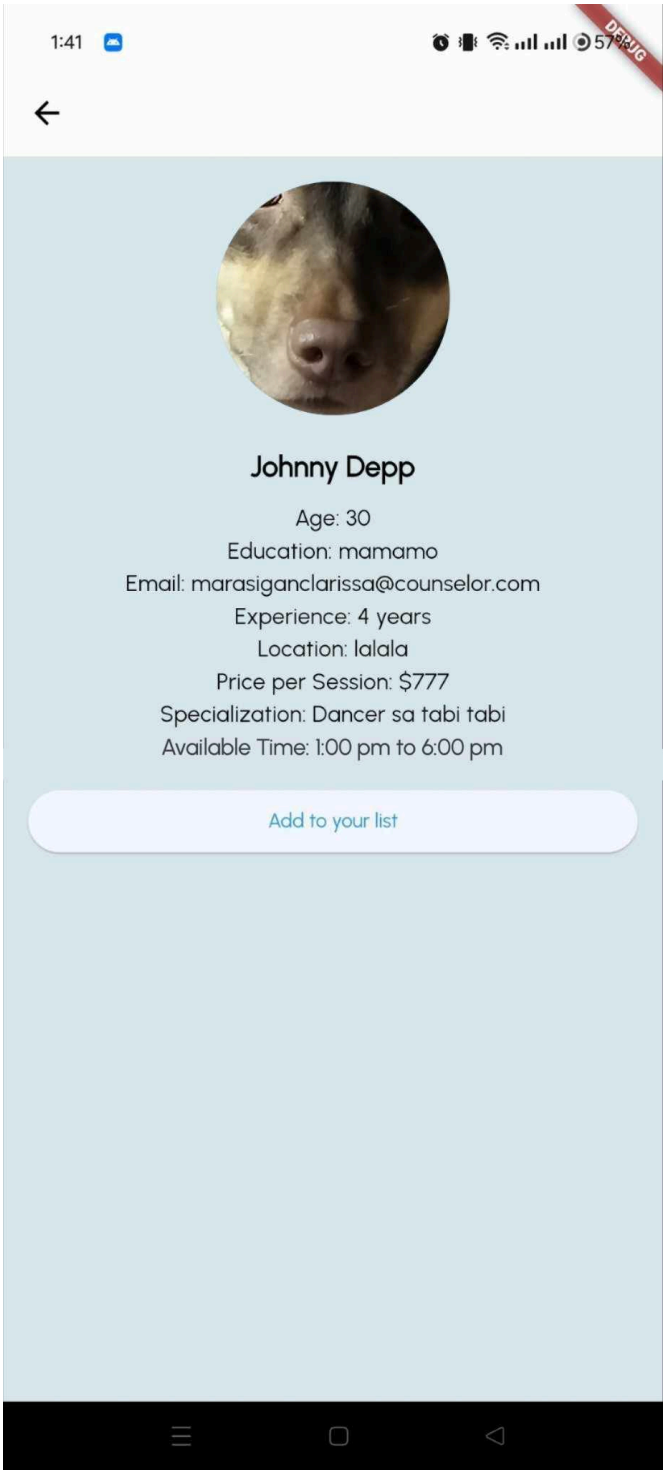


Figure 7 | Counselor Profile in Patients' Screen

This is the screen that will appear if the user/patient click on the counselor in Figure 6. In this screen, the user/patient will see the counselor's details such as their specialization, years of experience, and price per session. If the user/patient is satisfied with the counselor's profile, they can add the counselor by clicking the 'Add to your list' button.

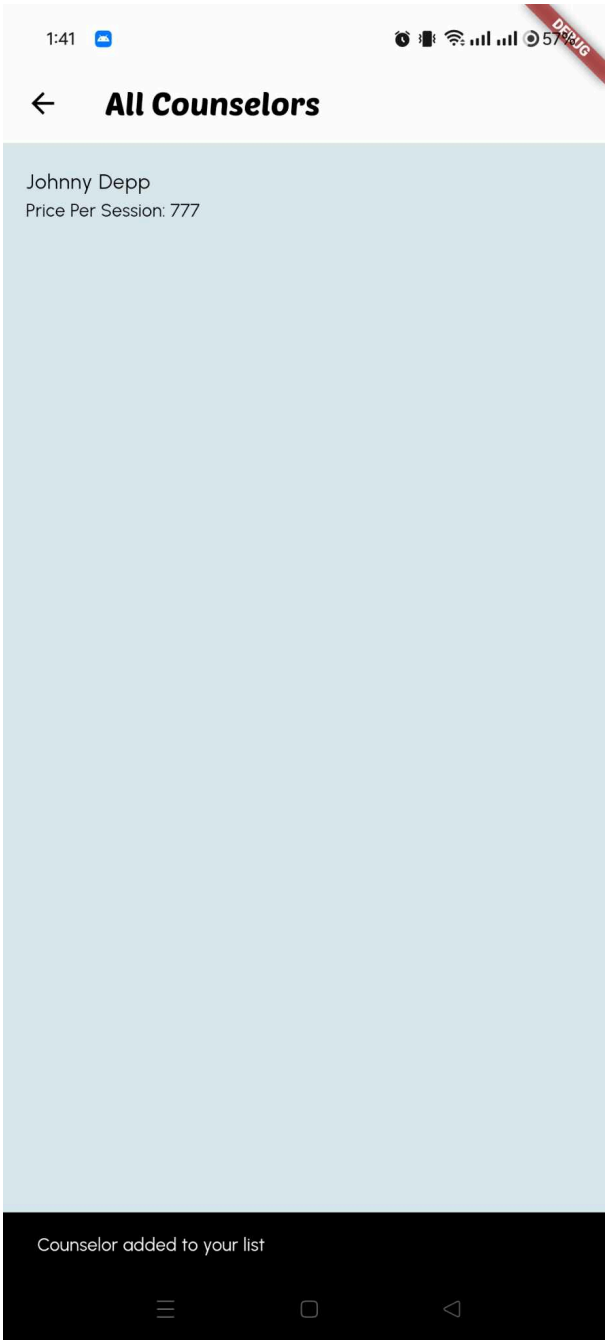


Figure 8 | Added notif

Once the counselor is added, the user/patient will be brought back to the same screen as Figure 6.

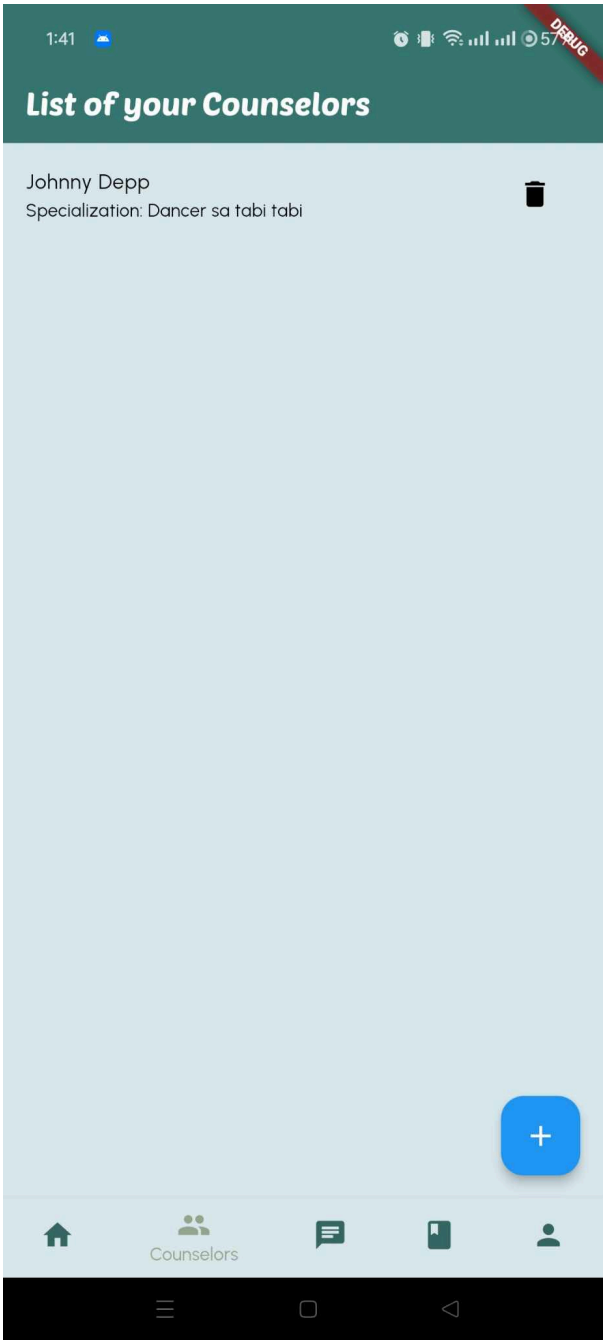


Figure 9 | Added list

When the user goes back to their personal list of counselors, the screen will show the user/patient that the counselor is already added to their list.

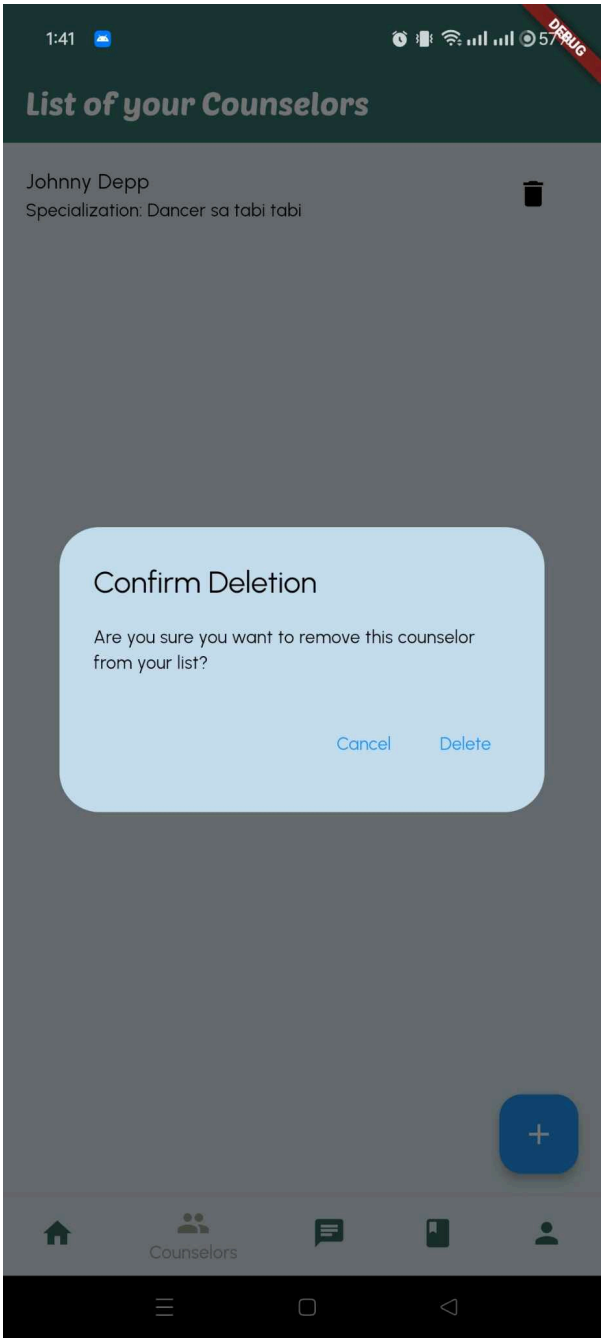


Figure 10 | Deletion Prompt

This prompt confirmation will appear if the user/patient clicks the trash bin button and wants to delete a counselor.

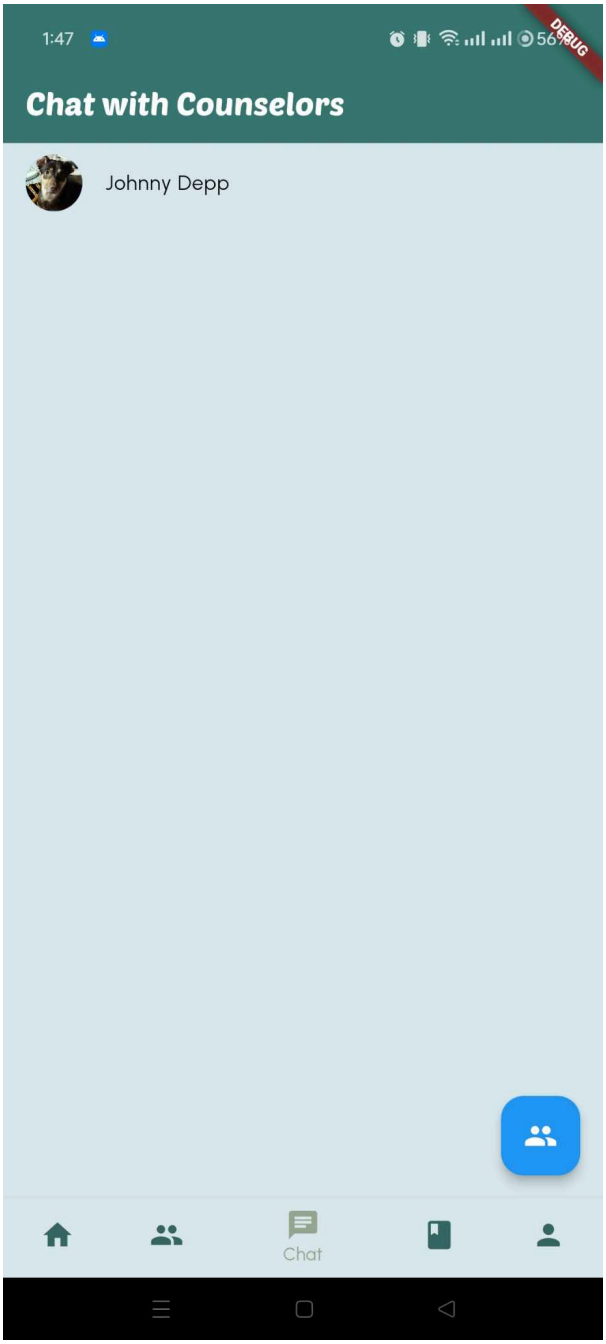


Figure 11 | Chat Page

This is the screen where the user/patient talks with the chosen counselors. The counselors will pop here once a conversation has started.

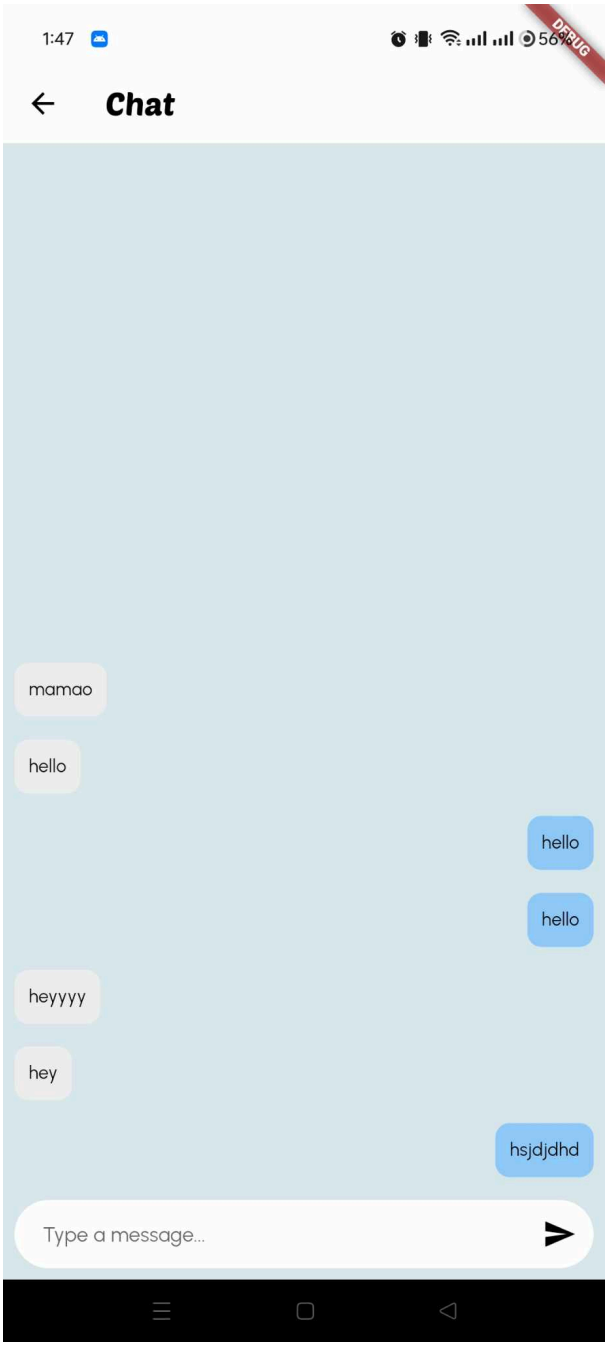


Figure 12 | Chatbox

This is what the chat box looks like when the user/patient clicks on one of the chats.

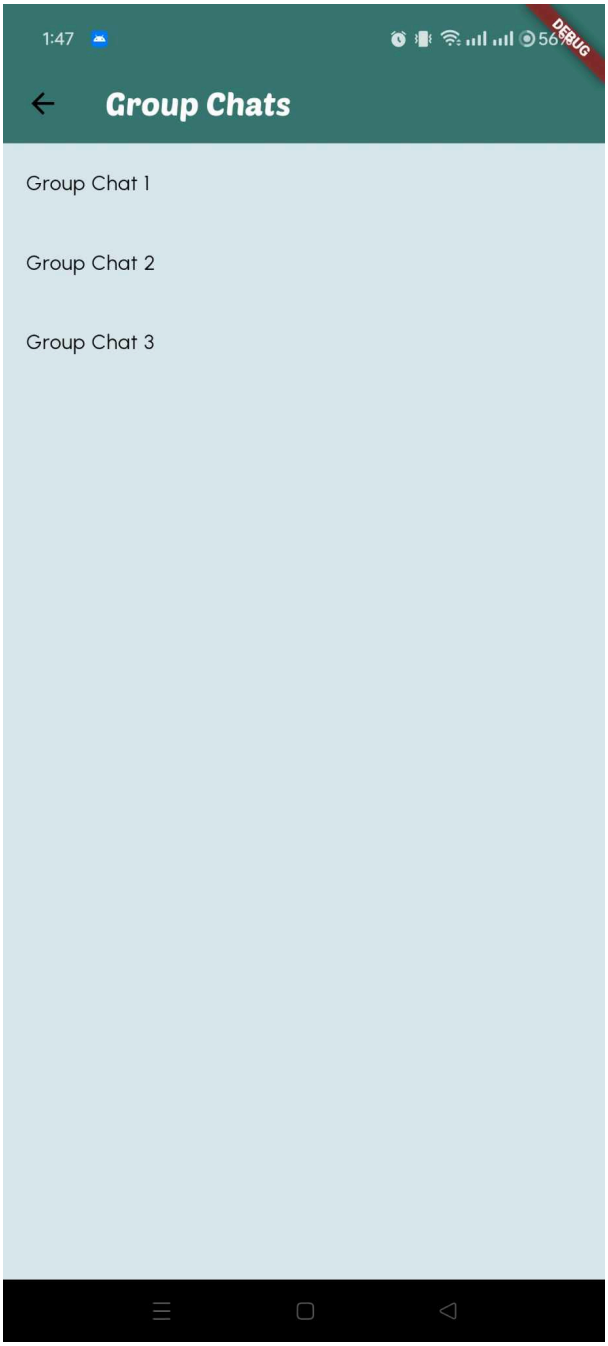


Figure 13 | Group Chat

This is where the user/patient can see the group chats that they are included in. This screen can also be seen on the Chat page.

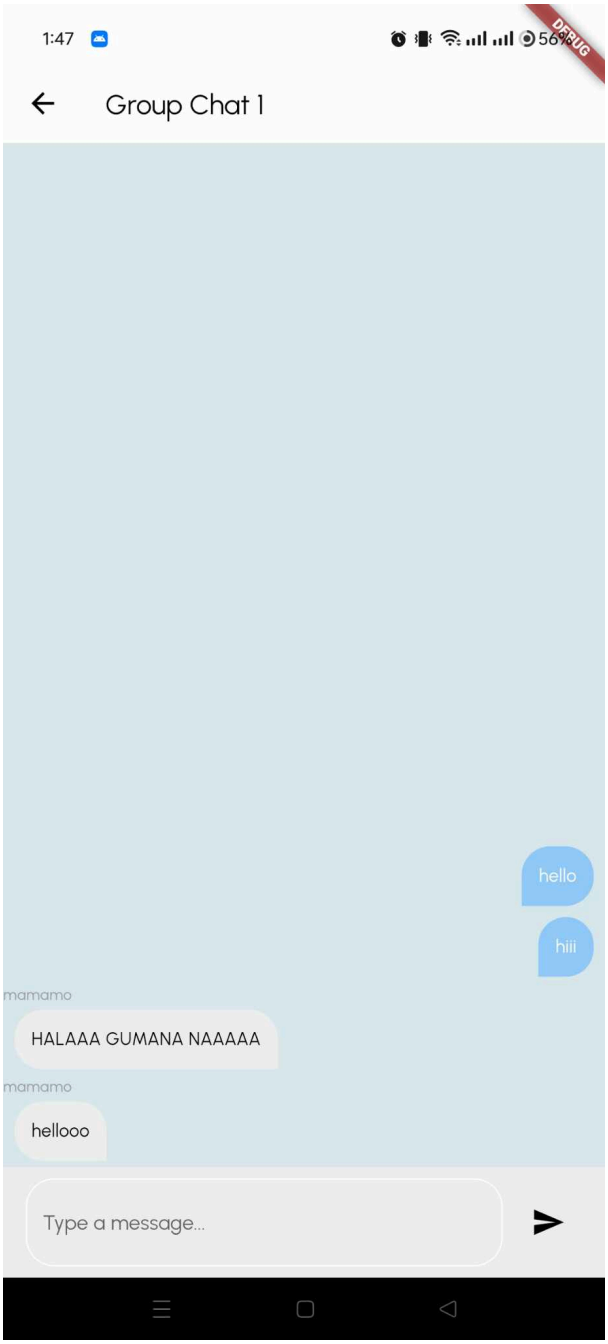


Figure 14 | Group Chatbox

This is what the group chat's chat box looks like when the user/patient clicks on one of the group chats.

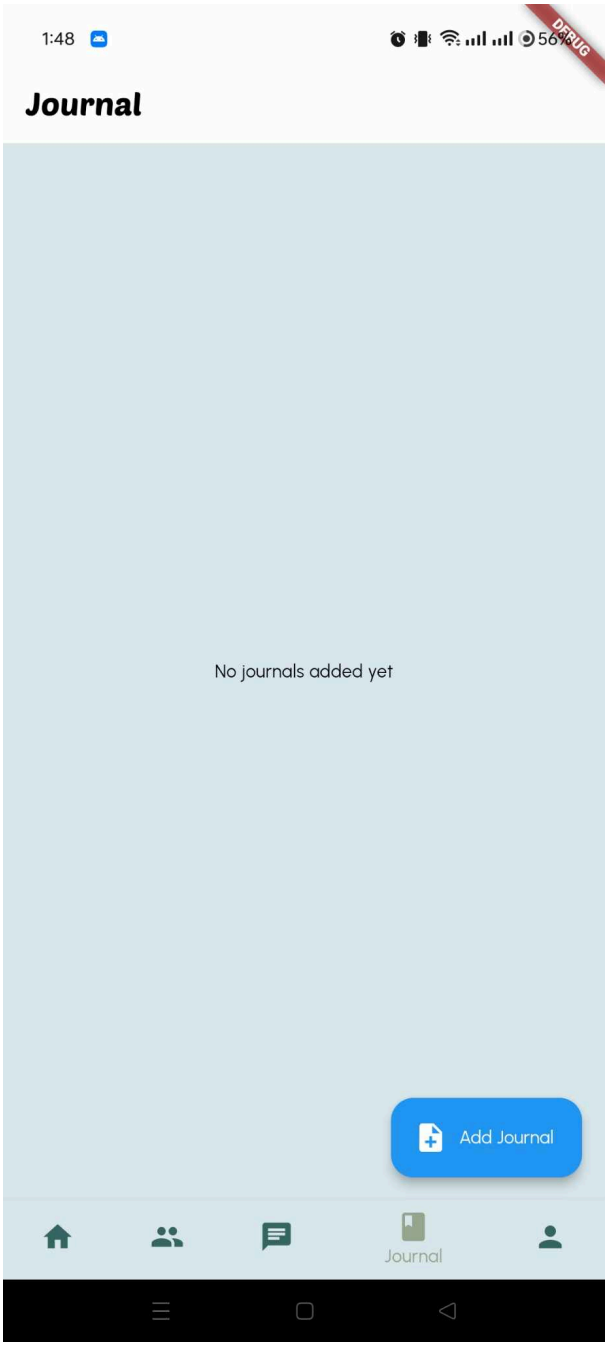


Figure 15 | Journal Page

This is the screen where the users/patients make their journals and see the saved journals that they've created in the past.

Figure 16 | Creating Journal Entry

If the user/patient wants to add a journal entry, this textbox will appear. Here, the user will write the title of their journal entry and may start writing anything in the text entry below the title.

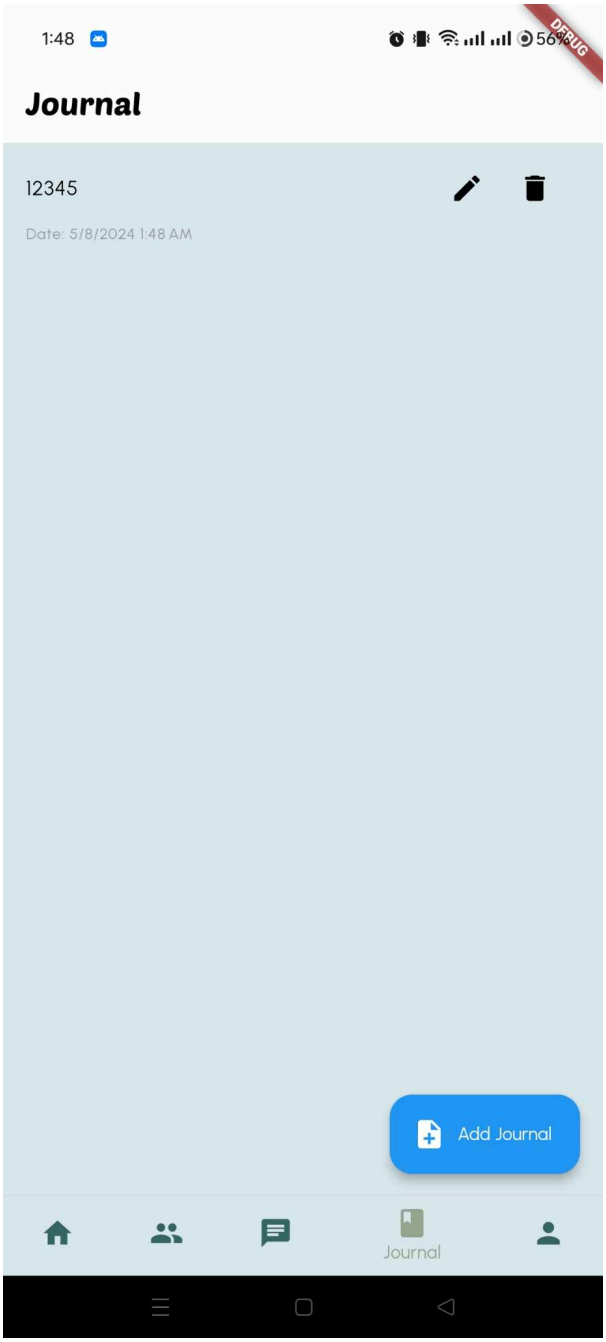


Figure 17 | Saved Journal Entry

Once the user/patient finishes their journal and saves it, the journal entry will then appear on the journal page.

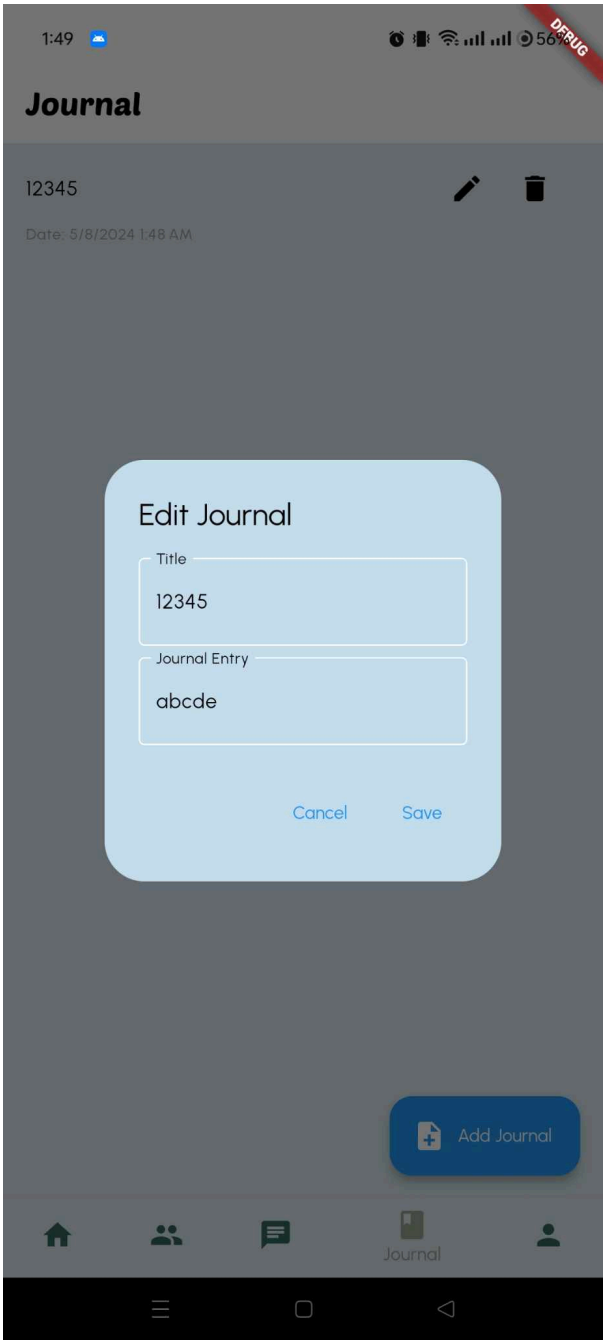


Figure 18 | Edit Journal

If the user/patient wants to edit the journal entry that they saved, they can click on the edit button and this textbox will appear. They can both edit the title of the journal and the content of it.

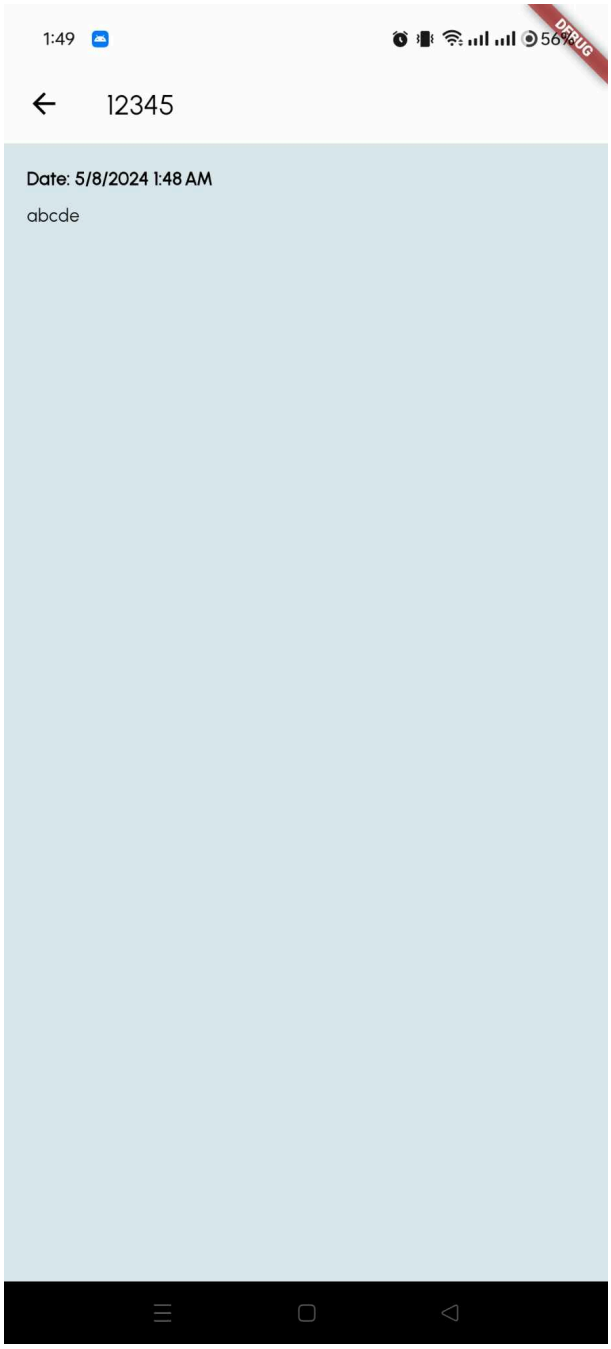


Figure 19 | Edited Journal

This is what the user will see after the edit that they've done.

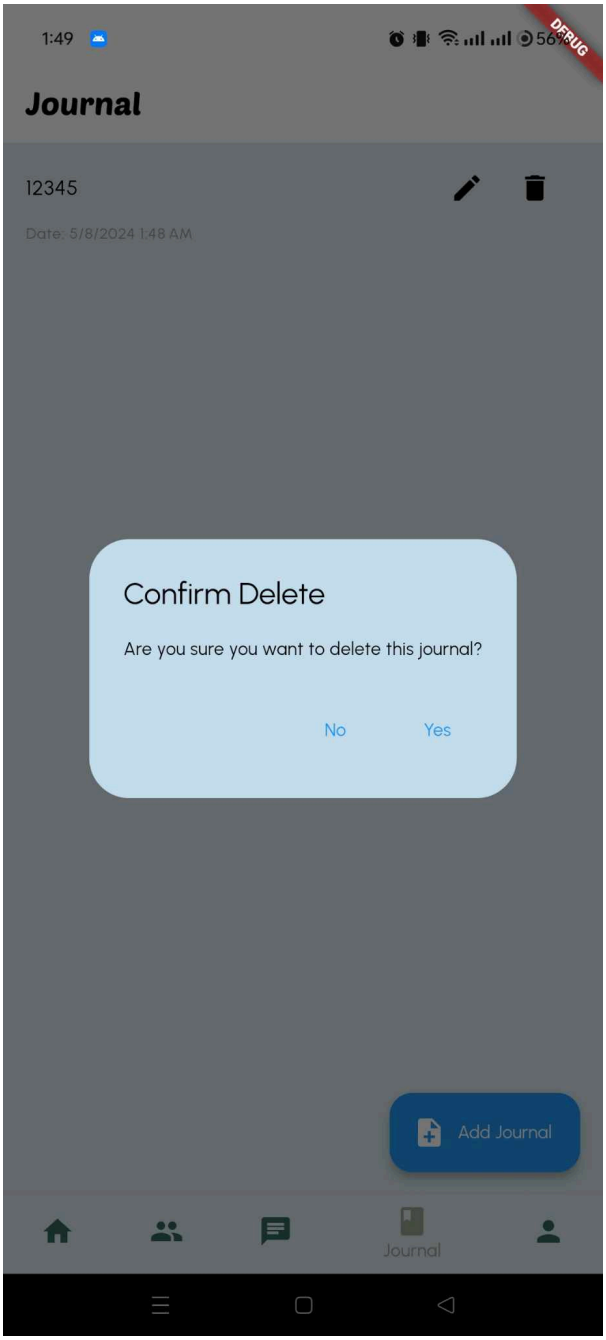


Figure 20 | Deletion Prompt

This prompt confirmation will appear if the user/patient clicks the trash bin button and wants to delete the journal entry.

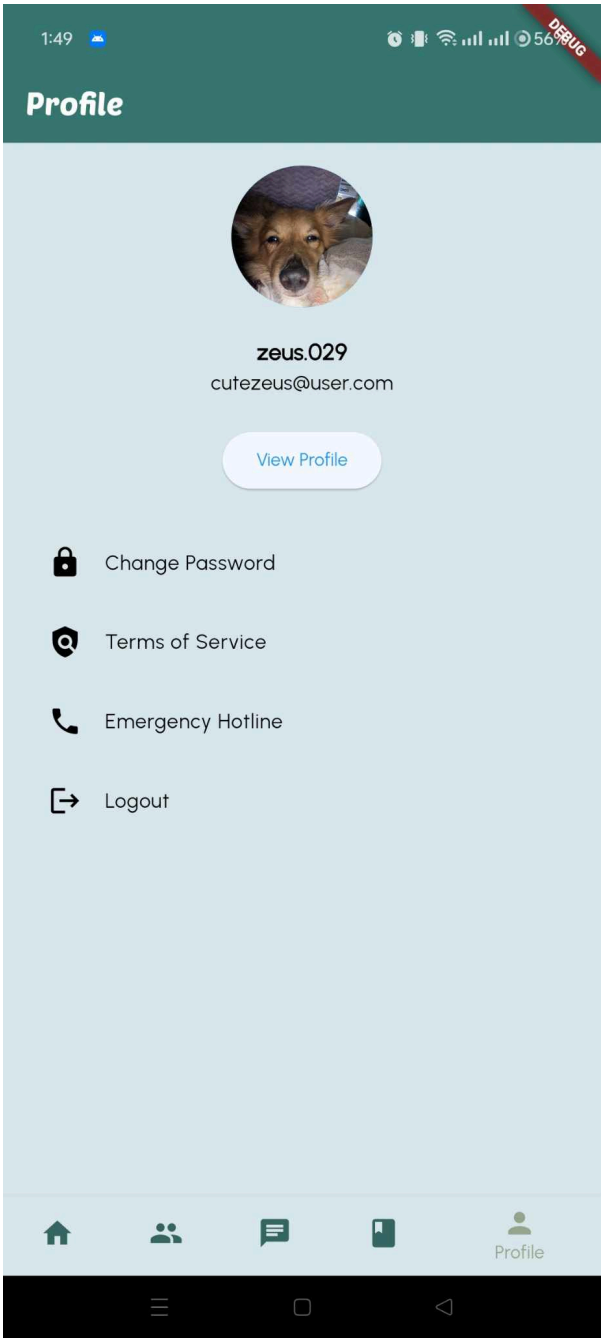


Figure 20 | User Profile Page

This is where the user/patient's profile page is. On this page, the user/patient can view their profile, change their password, see the terms and conditions of the application, and emergency hotline. The user/patient can also logout from their account here.

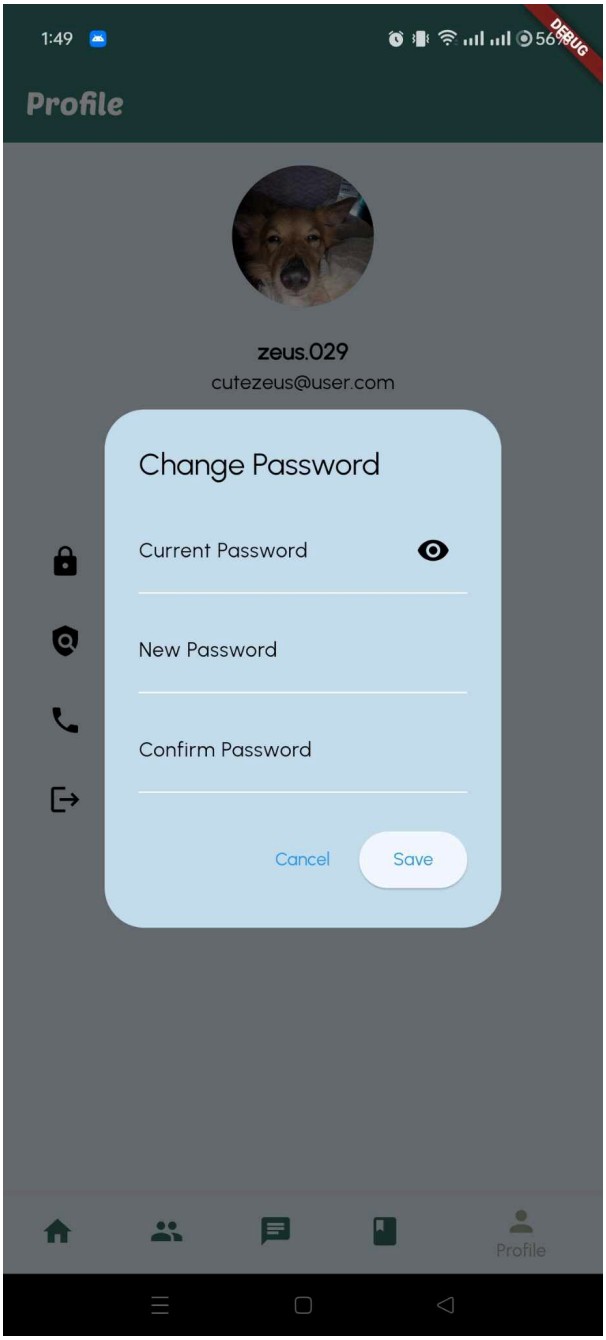


Figure 21 | Change Password

In this screen, if the user/patient chooses the change password in Figure 20, they will have to fill up and provide their current password, new password, and the confirmation password for them to be validated to change the account's password.

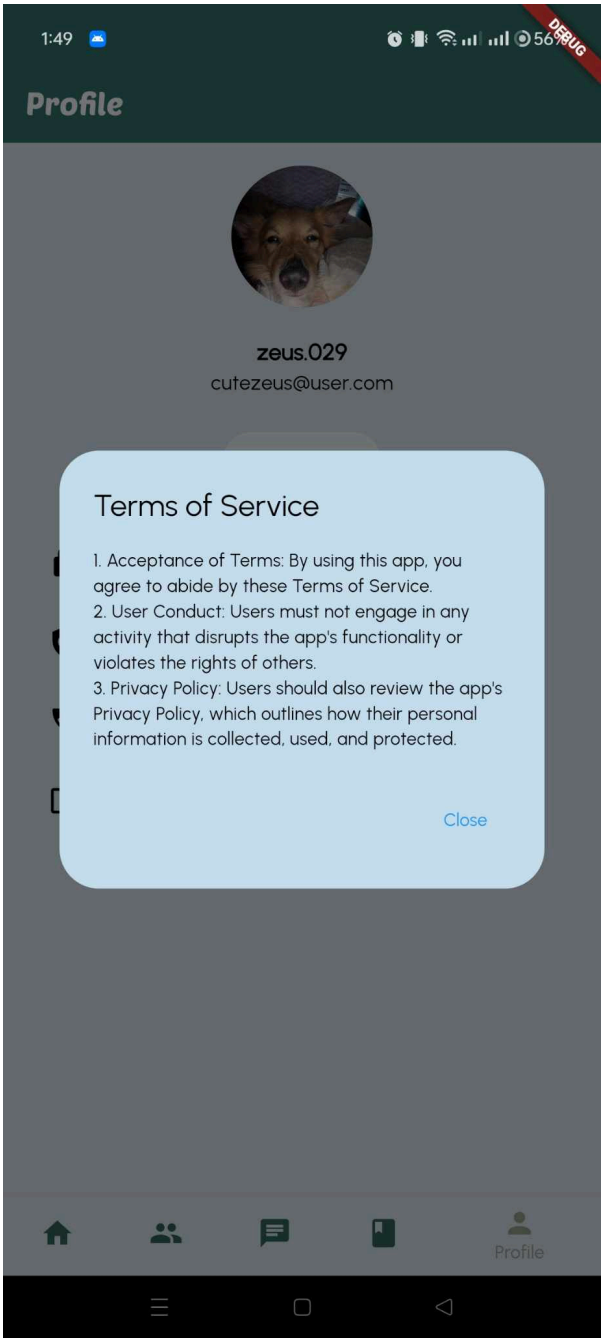


Figure 21 | Terms of Service

In this screen, if the user/patient chooses the Terms of Service in Figure 20, they will be able to read the terms of service of the application.

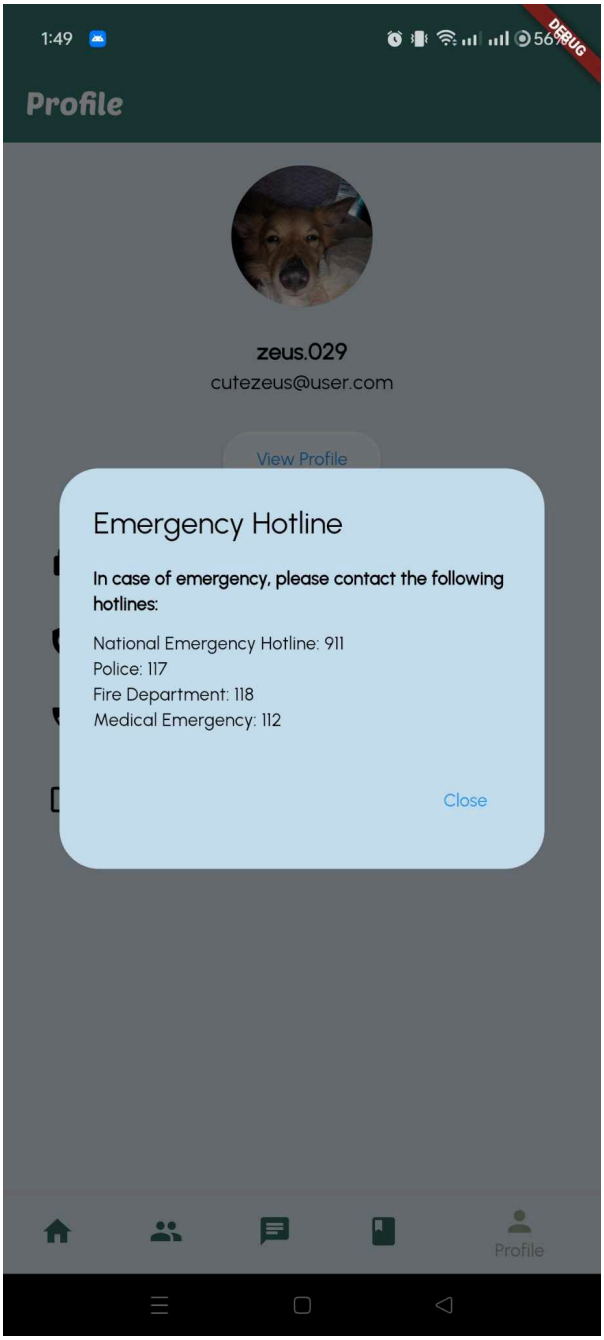


Figure 22 | Emergency Hotline

In this screen, if the user/patient chooses the Emergency Hotline in Figure 20, the application will provide them with the emergency hotline from various emergency services.

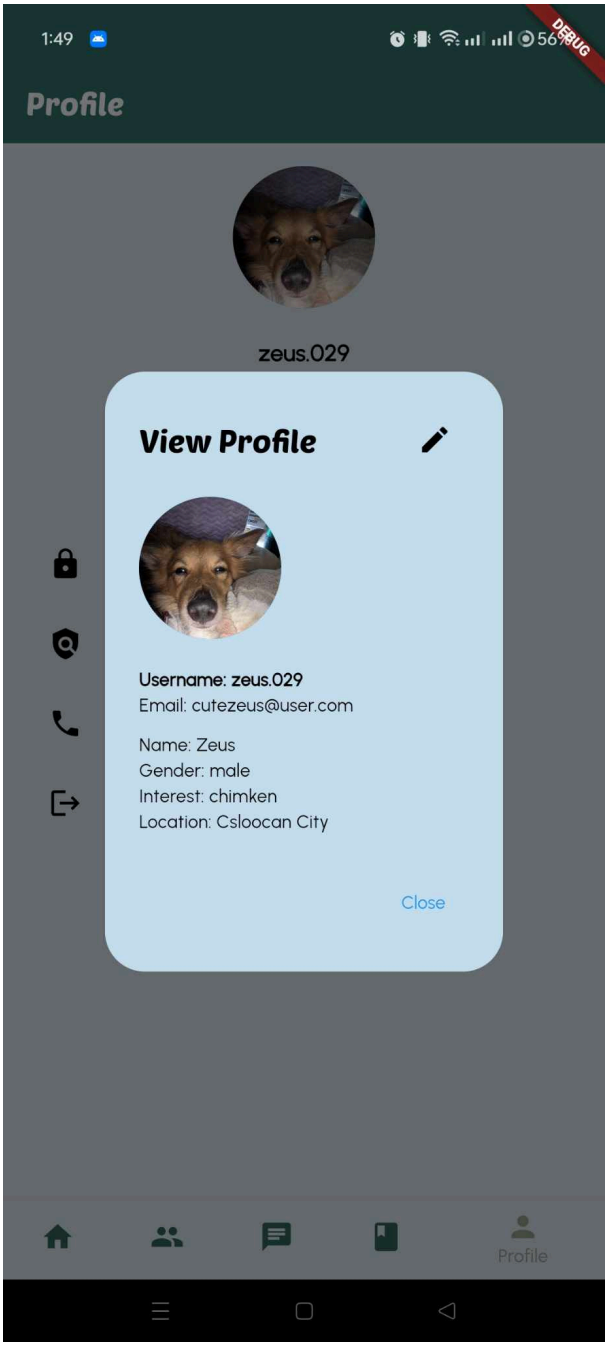


Figure 23 | Deletion Prompt

In this screen, if the user/patient chooses the view profile in Figure 20, they will be able to see the credentials such as the username, email, and the name, gender, interest, and location that they registered. There is also an edit button if the user/patient wants to edit their profile.

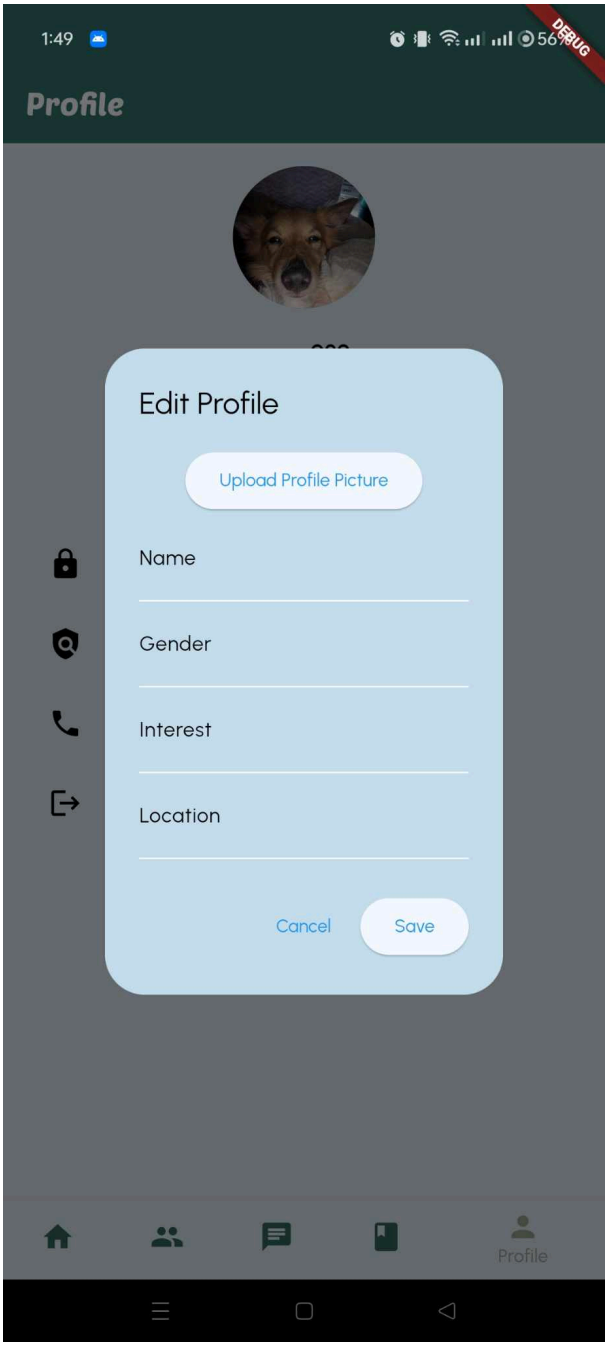


Figure 24 | Edit Profile

In this screen, if the user/patient wants to edit their profile, they will have to fill up the textbox that they want to edit, then click the save button to save the changes.

COUNSELOR SIDE

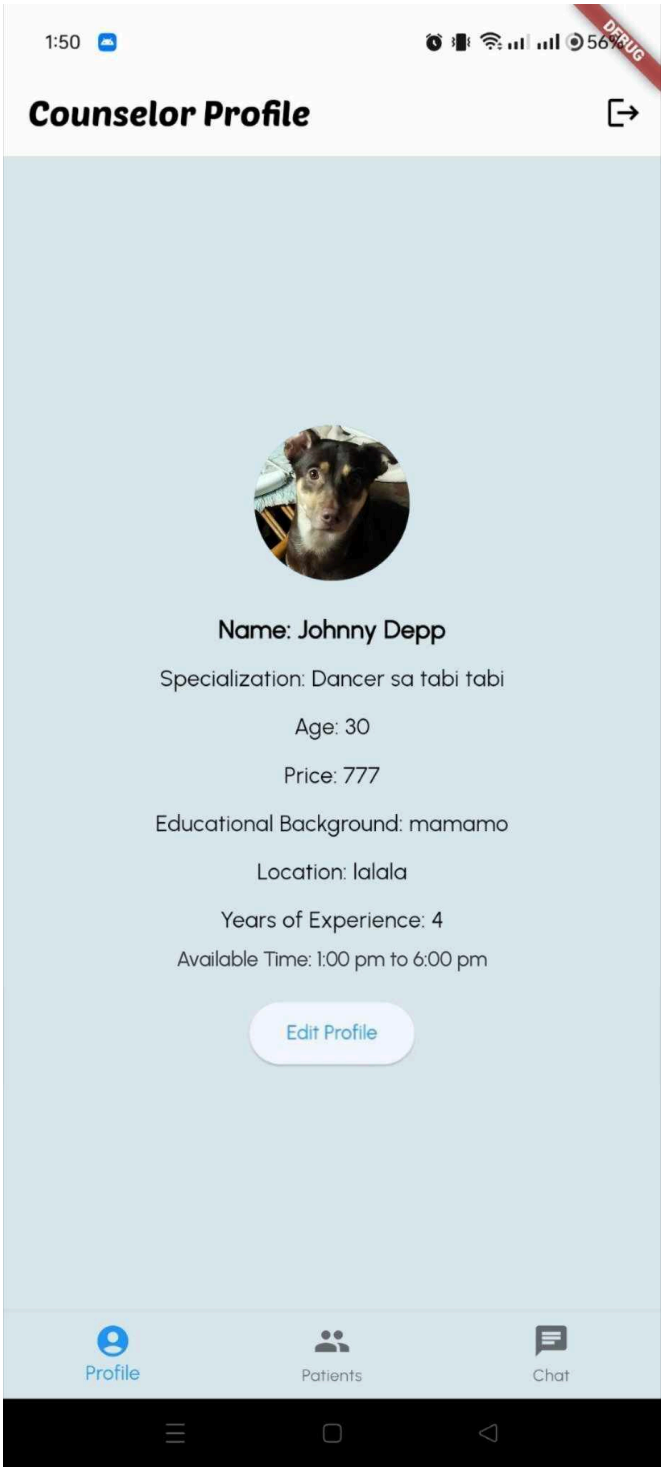


Figure 25 | Counselor Profile Page

This is where the counselor sees their profile and information. On this page, the counselor can view their general profile that the patients can see. There is also a button at the bottom where the counselor can edit the information about their profile information.

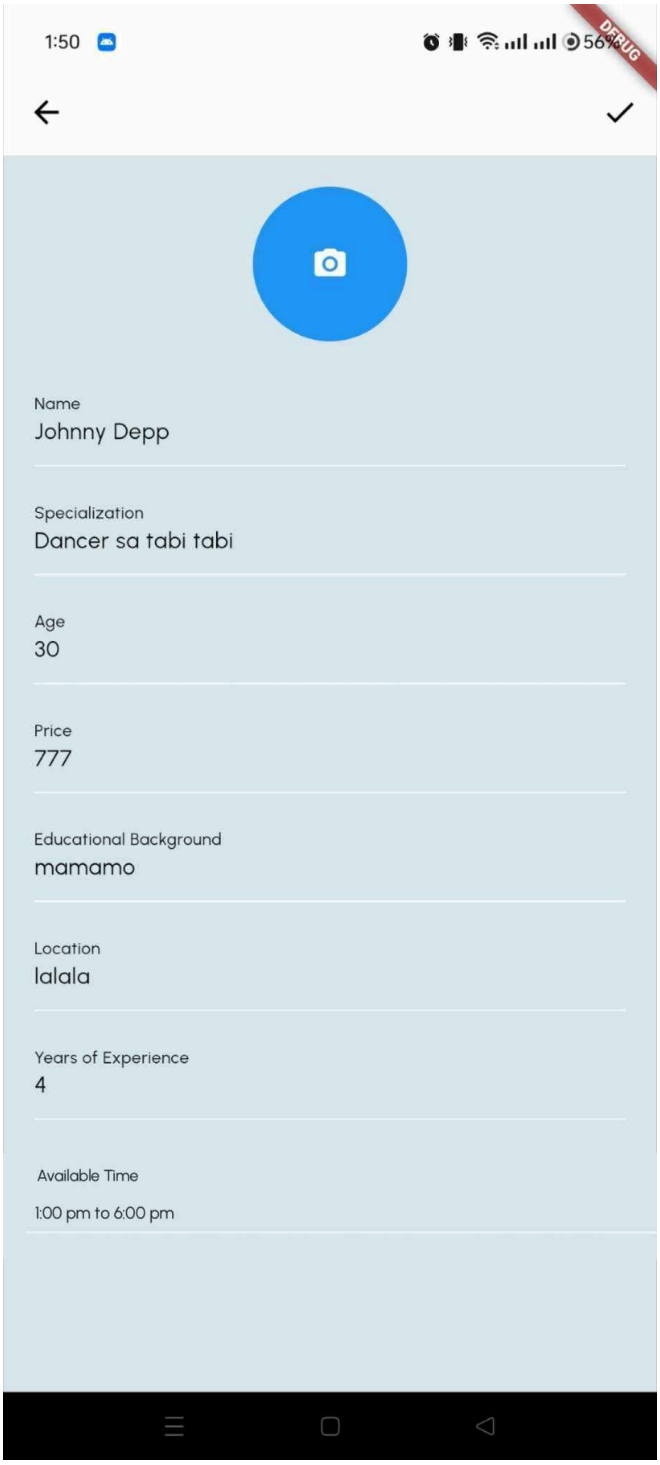


Figure 26 | Counselor Edit Profile

In this screen, if the counselor wants to edit their profile information, they will have to fill up the textbox that they want to edit, then click the save button to save the changes.

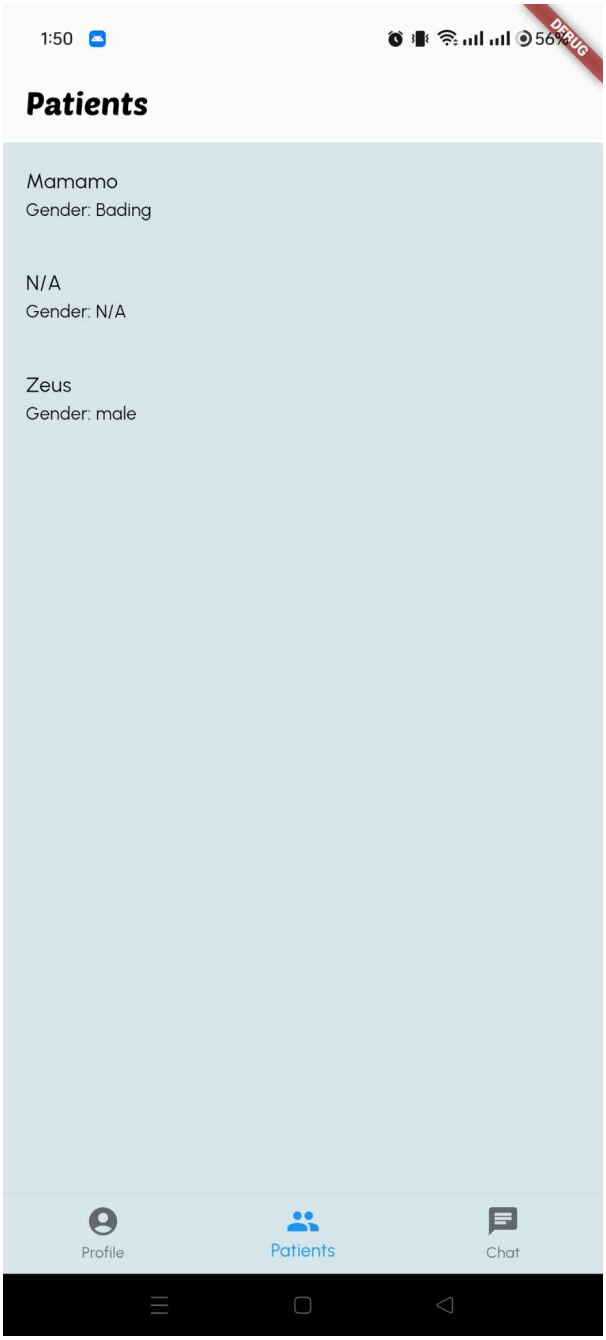


Figure 27 | Patient Page

This is the screen where the counselors can see and keep track of the patients that they have or the patients that chose/saved them. Here, they can see the patient's general information such as the patient's username and gender.

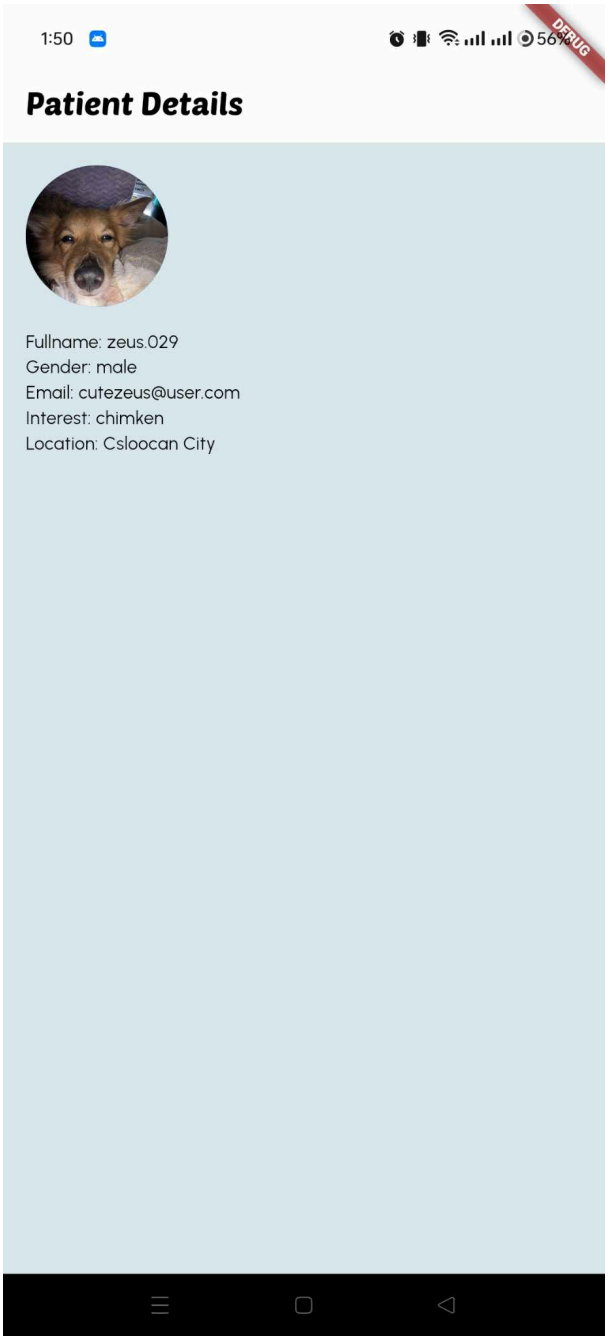


Figure 28 | Patient Details

Once the counselor clicks on one of the patients, they will be able to see the patient's full details.

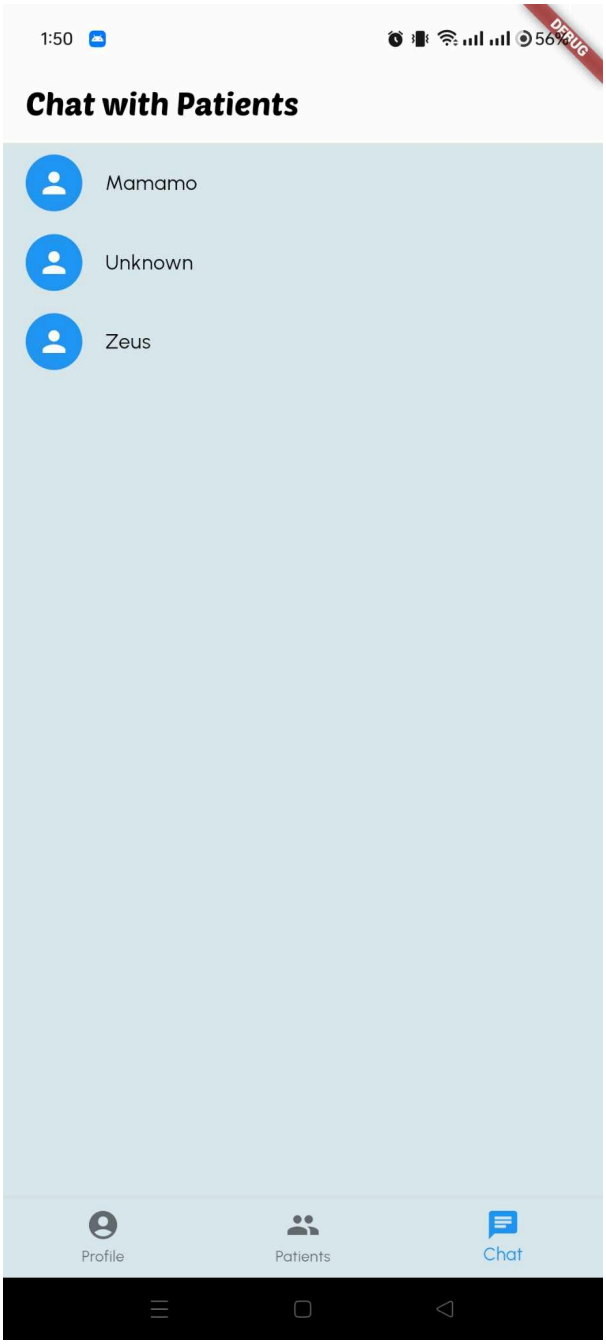


Figure 29 | Chat Page

This is the screen where the counselor talks with the patients who chose them. The patients will pop here once a conversation has started.

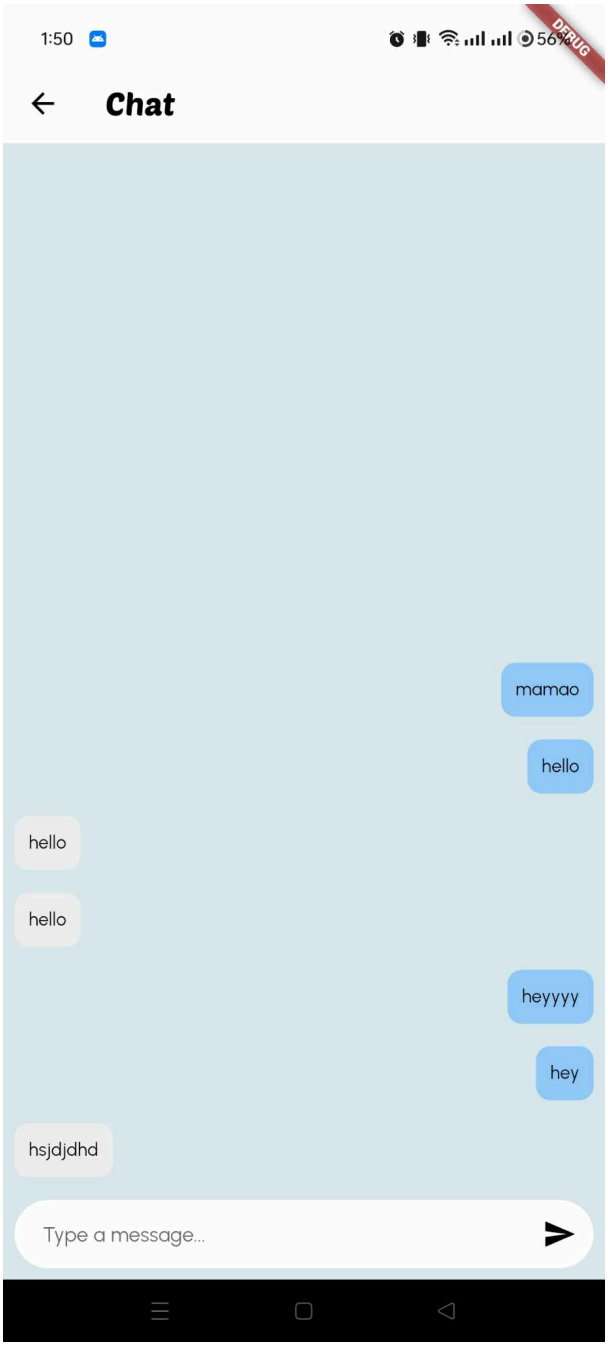


Figure 40 | Chatbox

This is what the chat box looks like when the counselor clicks on one of the chats.

Source Code

main .dart

```
import 'dart:developer';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'userscreen/signin_screen.dart';
import 'userscreen/home_screen.dart';
import 'userscreen/splash_screen.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  // Initialize Firebase
  try {
    await Firebase.initializeApp(
      options: const FirebaseOptions(
        apiKey: 'AIzaSyAO47B543CW0MpdoeLIfSMCgIwwf6nTI4k',
        appId: '1:730034838867:web:dc6f84fc6739116cb615a6',
        messagingSenderId: '730034838867',
        projectId: 'safespacedatabase',
        authDomain: 'safespacedatabase.firebaseapp.com',
        databaseURL:

'https://safespacedatabase-default-rtdb.asia-southeast1.firebasedatabase
.app',
        storageBucket: 'safespacedatabase.appspot.com',
      ),
    );
    log("Firebase initialized successfully!");
  } catch (e) {
    log("Error initializing Firebase: $e");
    // Handle Firebase initialization error
    // You can show an error message or take appropriate action here
  }

  runApp(const MyApp());
}
```

```dart
class MyApp extends StatelessWidget {
  const MyApp({Key? key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'SafeSpace',
      initialRoute: '/splash',
      routes: {
        '/splash': (context) => SplashScreen(),
        '/signin': (context) => LoginRegisterScreen(),
        '/home': (context) => HomeScreen(),
      },
      theme: ThemeData(
        colorScheme: ColorScheme.fromSwatch().copyWith(
          background: const Color(0xFFD9E8ED),
        ),
        fontFamily: 'urbanist',
      ),
    );
  }
}
```

Chatlist_screen.dart

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safespace/userscreen/counselor_screen.dart';
import 'package:safespace/userscreen/group_chat_screen.dart';
import 'package:safespace/userscreen/home_screen.dart';
import 'package:safespace/userscreen/journal_screen.dart';
import 'package:safespace/userscreen/messagescreen.dart';
import 'package:safespace/userscreen/profile_screen.dart'; // Import the
MessageScreen

class ChatScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        backgroundColor: Color(0xFF377671),
        title: Text(
          'Chat with Counselors',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
      ),
      body: StreamBuilder(
        stream:
FirebaseFirestore.instance.collection('counselors').snapshots(),
        builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          // Extract counselor data from Firestore snapshot and display
in a list
          return ListView.builder(
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              final counselor = snapshot.data!.docs[index];
              return ListTile(
                leading: CircleAvatar(
                  backgroundImage: NetworkImage(
                      counselor['profileImageUrl'] ??
'default_image_url'),
                ),
                title: Text(counselor['fullname'] ?? 'Anonymous'),
                onTap: () {
                  Navigator.push(
```

```
                    context,
                    MaterialPageRoute(
                      builder: (context) => MessageScreen(
                        userId: FirebaseAuth.instance.currentUser?.uid
?? "",
                        userType:
                            'user', // Indicate that the user is
messaging the counselor
                        recipientId: counselor
                            .id, // The recipientId is the counselor's
ID
                      ),
                    ),
                  );
                },
              );
            },
          );
        },
      ),
      bottomNavigationBar: BottomNavigationBar(
        backgroundColor: const Color(0xFF377671),
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Counselors',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.book),
            label: 'Journal',
          ),
```

```
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        unselectedItemColor: Color(0xFF366763),
        selectedItemColor: Color.fromARGB(255, 150, 167, 144),
        currentIndex: 2, // Set the current index to 2 for the Chat
screen
        onTap: (index) {
          switch (index) {
            case 0:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => HomeScreen()),
              );
              break;
            case 1:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
CounselorScreen()),
              );
              break;
            case 3:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
JournalScreen()),
              );
              break;
            case 4:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
ProfileScreen()),
              );
              break;
          }
```

```
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          // Navigate to group chat screen when floating action button
is pressed
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => GroupChatScreen()),
          );
        },
        child: Icon(Icons.group), // Group chat icon
      ),
    );
  }
}
```

Counselor_editprofile_screen.dart

```
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart'; // Import
Firebase Storage
import 'package:image_picker/image_picker.dart'; // Import ImagePicker

class EditProfileScreen extends StatefulWidget {
  @override
  _EditProfileScreenState createState() => _EditProfileScreenState();
}

class _EditProfileScreenState extends State<EditProfileScreen> {
  late User? _user;
  late TextEditingController _nameController;
  late TextEditingController _specializationController;
  late TextEditingController _ageController;
  late TextEditingController _priceController;
```

```
  late TextEditingController _educationController;
  late TextEditingController _locationController;
  late TextEditingController _experienceController;
  File? _image;

  @override
  void initState() {
    super.initState();
    _user = FirebaseAuth.instance.currentUser;
    _nameController = TextEditingController();
    _specializationController = TextEditingController();
    _ageController = TextEditingController();
    _priceController = TextEditingController();
    _educationController = TextEditingController();
    _locationController = TextEditingController();
    _experienceController = TextEditingController();
    _fetchCounselorData();
  }

  @override
  void dispose() {
    _nameController.dispose();
    _specializationController.dispose();
    _ageController.dispose();
    _priceController.dispose();
    _educationController.dispose();
    _locationController.dispose();
    _experienceController.dispose();
    super.dispose();
  }

  Future<void> _fetchCounselorData() async {
    if (_user != null) {
      DocumentSnapshot snapshot = await FirebaseFirestore.instance
          .collection('counselors')
          .doc(_user!.uid)
          .get();
      Map<String, dynamic> data = snapshot.data() as Map<String,
dynamic>;
```

```
      setState(() {
        _nameController.text = data['fullname'] ?? '';
        _specializationController.text = data['specialization'] ?? '';
        _ageController.text = data['age'] ?? '';
        _priceController.text = data['price'] ?? '';
        _educationController.text = data['education'] ?? '';
        _locationController.text = data['location'] ?? '';
        _experienceController.text = data['experience'] ?? '';
      });
    }
  }

  Future<void> _updateProfile() async {
    print('Updating profile...');
    if (_user != null) {
      print('User is not null');

      Map<String, dynamic> profileData = {
        'fullname': _nameController.text,
        'specialization': _specializationController.text,
        'age': _ageController.text,
        'price': _priceController.text,
        'education': _educationController.text,
        'location': _locationController.text,
        'experience': _experienceController.text,
      };

      if (_image != null) {
        print('Uploading image...');
        String imageUrl = await _uploadImage();
        if (imageUrl.isNotEmpty) {
          profileData['profileImageUrl'] = imageUrl;
        } else {
          // Handle case when image upload fails
          print('Failed to upload image');
          return;
        }
      }
```

```
      print('Updating profile data...');
      await FirebaseFirestore.instance
          .collection('counselors')
          .doc(_user!.uid)
          .update(profileData);
      print('Profile updated successfully');

      print('Navigating back to previous screen...');
      Navigator.pop(context); // Return to the previous screen
    }
  }

  Future<String> _uploadImage() async {
    String imageUrl = '';

    try {
      String fileName = _user!.uid;
      Reference reference = FirebaseStorage.instance
          .ref()
          .child('profileImages/$fileName.jpg'); // Use Reference
      UploadTask uploadTask = reference.putFile(_image!);
      TaskSnapshot taskSnapshot = await uploadTask; // Use TaskSnapshot
      imageUrl = await taskSnapshot.ref.getDownloadURL();
      print('Image uploaded successfully. URL: $imageUrl');
    } catch (e) {
      print('Error uploading image: $e');
    }

    return imageUrl;
  }

  Future<void> _getImageFromGallery() async {
    final picker = ImagePicker();
    final pickedFile = await picker.pickImage(source:
ImageSource.gallery);

    print(
        'Image picked from gallery'); // Add this line to ensure the
function is called
```

```
    setState(() {
      if (pickedFile != null) {
        _image = File(pickedFile.path);
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'Edit Profile',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.done),
            onPressed: _updateProfile,
          ),
        ],
      ),
      body: SingleChildScrollView(
        padding: EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Center(
              child: GestureDetector(
                onTap: _getImageFromGallery,
                child: CircleAvatar(
                  radius: 50,
                  backgroundImage: _image != null ? FileImage(_image!) :
null,
```

```
                  child: _image == null ? Icon(Icons.camera_alt) : null,
                ),
              ),
            ),
            SizedBox(height: 20),
            TextField(
              controller: _nameController,
              decoration: InputDecoration(labelText: 'Name'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _specializationController,
              decoration: InputDecoration(labelText: 'Specialization'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _ageController,
              decoration: InputDecoration(labelText: 'Age'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _priceController,
              decoration: InputDecoration(labelText: 'Price'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _educationController,
              decoration: InputDecoration(labelText: 'Educational
Background'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _locationController,
              decoration: InputDecoration(labelText: 'Location'),
            ),
            SizedBox(height: 10),
            TextField(
              controller: _experienceController,
              decoration: InputDecoration(labelText: 'Years of
```

```
Experience'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Counselor_profile_screen.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safespace/userscreen/counselor_editprofile_screen.dart';
import 'package:safespace/userscreen/counselor_patient_screen.dart';
import 'package:safespace/userscreen/counselorchatlist_screen.dart';
import 'package:safespace/userscreen/signin_screen.dart'; // Import your
sign-in screen file here

class CounselorHomeScreen extends StatefulWidget {
  @override
  _CounselorHomeScreenState createState() =>
_CounselorHomeScreenState();
}

class _CounselorHomeScreenState extends State<CounselorHomeScreen> {
  late User? _user;
  late Map<String, dynamic>? _counselorData;

  @override
  void initState() {
    super.initState();
    _user = FirebaseAuth.instance.currentUser;
    _fetchCounselorData();
  }

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: const Text(
          'Counselor Profile',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
        actions: [
          IconButton(
            icon: Icon(Icons.logout),
            onPressed: _logout,
          ),
        ],
      ),
      body: GestureDetector(
        onTap: _editProfile, // Call edit profile function when tapped
        child: Center(
          child: _counselorData != null
              ? SingleChildScrollView(
                  child: Padding(
                    padding: const EdgeInsets.all(20.0),
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.center,
                      crossAxisAlignment: CrossAxisAlignment.center,
                      children: [
                        CircleAvatar(
                          radius: 50,
                          backgroundImage: NetworkImage(
                              _counselorData!['profileImageUrl'] ?? ''),
                        ),
                        const SizedBox(height: 20),
                        Text(
                          'Name: ${_counselorData!['fullname'] ??
'Counselor Name'}',
                          style: const TextStyle(
```

```
                            fontSize: 18,
                            fontWeight: FontWeight.bold,
                          ),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Specialization:
${_counselorData!['specialization'] ?? 'Psychiatry'}',
                          style: const TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Age: ${_counselorData!['age'] ?? 'Not
specified'}',
                          style: const TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Price: ${_counselorData!['price'] ?? 'Not
specified'}',
                          style: const TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Educational Background:
${_counselorData!['education'] ?? 'Not specified'}',
                          style: const TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Location: ${_counselorData!['location'] ??
'Not specified'}',
                          style: const TextStyle(fontSize: 16),
                        ),
                        const SizedBox(height: 10),
                        Text(
                          'Years of Experience:
${_counselorData!['experience'] ?? 'Not specified'}',
                          style: const TextStyle(fontSize: 16),
```

```
                      ),
                      const SizedBox(height: 20),
                      ElevatedButton(
                        onPressed: _editProfile,
                        child: const Text('Edit Profile'),
                      ),
                    ],
                  ),
                ),
              )
            : const CircularProgressIndicator(),
      ),
    ),
    bottomNavigationBar: BottomNavigationBar(
      items: const <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: 'Profile',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.people),
          label: 'Patients',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.chat),
          label: 'Chat',
        ),
      ],
      currentIndex: 0, // Assuming Profile is the first item
      selectedItemColor: Colors.blue,
      onTap: _onItemTapped,
    ),
  );
}

Future<void> _fetchCounselorData() async {
  if (_user != null) {
    DocumentSnapshot snapshot = await FirebaseFirestore.instance
        .collection('counselors')
```

```
          .doc(_user!.uid)
          .get();
      if (snapshot.data() != null) {
        setState(() {
          _counselorData = snapshot.data() as Map<String, dynamic>;
        });
      }
    }
  }

  void _editProfile() {
    // Navigate to edit profile screen
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => EditProfileScreen()),
    );
  }

  void _logout() async {
    await FirebaseAuth.instance.signOut();
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(
          builder: (context) =>
              const LoginRegisterScreen()), // Replace SignInScreen()
with your sign-in screen route
    );
  }

  void _onItemTapped(int index) {
    switch (index) {
      case 0:
        // Navigate to counselor profile screen (current screen)
        break;
      case 1:
        // Navigate to patient screen
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => PatientScreen()),
```

```
        );
        break;
      case 2:
        // Navigate to chat screen
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) =>
CounselorChatScreen()),
        );
        break;
    }
  }
}
```

Counselor_patient_screen.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safespace/userscreen/counselor_home_screen.dart';
import 'package:safespace/userscreen/counselorchatlist_screen.dart';
import 'package:safespace/userscreen/patientdetailscreen.dart';

class PatientScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: Text(
          'Patients',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
      ),
      body: StreamBuilder(
```

```
        stream: FirebaseFirestore.instance
            .collection('counselors')
            .doc(getCurrentUserId())
            .collection('patients')
            .snapshots(),
        builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          // Display list of patients
          return ListView.builder(
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              final patientUserId = snapshot.data!.docs[index].id;
              return FutureBuilder(
                future: FirebaseFirestore.instance
                    .collection('users')
                    .doc(patientUserId)
                    .get(),
                builder:
                    (context, AsyncSnapshot<DocumentSnapshot>
userSnapshot) {
                  if (userSnapshot.connectionState ==
ConnectionState.waiting) {
                    return ListTile(
                      title: Text('Loading...'),
                    );
                  }
                  if (userSnapshot.hasError) {
                    return ListTile(
                      title: Text('Error: ${userSnapshot.error}'),
                    );
                  }
                  if (!userSnapshot.hasData || userSnapshot.data ==
null) {
```

```
                    return ListTile(
                      title: Text('User not found'),
                    );
                  }
                  final patientData =
                      userSnapshot.data?.data() as Map<String, dynamic>?
?? {};
                  final fullname = patientData['name'] as String? ??
'N/A';
                  final gender = patientData['gender'] as String? ??
'N/A';
                  return ListTile(
                    title: Row(
                      children: [
                        Expanded(
                          child: Text(fullname),
                        ),
                      ],
                    ),
                    subtitle: Text('Gender: $gender'),
                    onTap: () {
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                            builder: (context) =>
                                PatientDetailsScreen(patientUserId)),
                      );
                    },
                  );
                },
              );
            },
          );
        },
      ),
      bottomNavigationBar: BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.account_circle),
```

```
            label: 'Profile',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Patients',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
        ],
        currentIndex: 1, // Assuming Patients is the current screen
        selectedItemColor: Colors.blue,
        onTap: (index) {
          String patientUserId =
              getCurrentUserId(); // Obtain patientUserId here
          _onItemTapped(context, index, patientUserId);
        },
      ),
    );
  }

  void _onItemTapped(BuildContext context, int index, String
patientUserId) {
    switch (index) {
      case 0:
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) =>
CounselorHomeScreen()),
        );
        break;
      case 1:
        // No action needed as the user is already on the Patients
screen
        break;
      case 2:
        Navigator.push(
          context,
```

```
          MaterialPageRoute(
            builder: (context) => CounselorChatScreen(),
          ),
        );

        break;
    }
  }

  String getCurrentUserId() {
    // Get the current user's ID from FirebaseAuth
    User? user = FirebaseAuth.instance.currentUser;
    // Check if a user is signed in
    if (user != null) {
      // User is signed in, return their user ID
      return user.uid;
    } else {
      // No user signed in, return an empty string or handle as needed
      return '';
    }
  }
}
```

Counselor_screen.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safespace/userscreen/chatlist_screen.dart';
import 'package:safespace/userscreen/home_screen.dart';
import 'package:safespace/userscreen/journal_screen.dart';
import 'package:safespace/userscreen/profile_screen.dart';

class CounselorScreen extends StatefulWidget {
  @override
  _CounselorScreenState createState() => _CounselorScreenState();
}

class _CounselorScreenState extends State<CounselorScreen> {
```

```dart
late Stream<QuerySnapshot> _userCounselorsStream;

@override
void initState() {
  super.initState();
  _loadUserCounselors();
}

void _loadUserCounselors() {
  String? userId = getCurrentUserId();
  _userCounselorsStream = FirebaseFirestore.instance
      .collection('users')
      .doc(userId)
      .collection('professional_counselors')
      .snapshots();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      automaticallyImplyLeading: false,
      backgroundColor: Color(0xFF377671),
      title: Text(
        'List of your Counselors',
        style: TextStyle(
          fontFamily: 'poetsen',
          color: Colors.white,
          fontSize: 25,
        ),
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        _navigateToAllCounselors(context);
      },
      child: Icon(Icons.add),
    ),
    bottomNavigationBar: BottomNavigationBar(
```

```
        backgroundColor: const Color(0xFF377671),
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Counselors',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.book),
            label: 'Journal',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        unselectedItemColor: Color(0xFF366763),
        selectedItemColor: Color.fromARGB(255, 150, 167, 144),
        currentIndex: 1,
        onTap: (index) {
          switch (index) {
            case 0:
              _navigateTo(context, HomeScreen());
              break;
            case 1:
              // No need to navigate, already on counselor screen
              break;
            case 2:
              _navigateTo(context, ChatScreen());
              break;
            case 3:
              _navigateTo(context, JournalScreen());
```

```
              break;
            case 4:
              _navigateTo(context, ProfileScreen());
              break;
          }
        },
      ),
      body: StreamBuilder(
        stream: _userCounselorsStream,
        builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          // Display the list of user's counselors
          if (snapshot.data!.docs.isEmpty) {
            return Center(child: Text('You have no counselors yet.'));
          }
          return ListView.builder(
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              final counselor = snapshot.data!.docs[index];
              return ListTile(
                title: Text(counselor['fullname']),
                subtitle: Text(
                    'Specialization: ${counselor['specialization'] ??
'N/A'}'),
                trailing: IconButton(
                  icon: Icon(Icons.delete),
                  onPressed: () {
                    showDialog(
                      context: context,
                      builder: (BuildContext context) {
                        return AlertDialog(
                          title: Text('Confirm Deletion'),
                          content: Text(
```

```
                              'Are you sure you want to remove this
counselor from your list?'),
                          actions: <Widget>[
                            TextButton(
                              child: Text('Cancel'),
                              onPressed: () {
                                Navigator.of(context).pop();
                              },
                            ),
                            TextButton(
                              child: Text('Delete'),
                              onPressed: () {
                                _removeFromUserList(context, counselor);
                                Navigator.of(context).pop();
                              },
                            ),
                          ],
                        );
                      },
                    );
                  },
                ),
              );
            },
          );
        },
      ),
    );
  }

  void _navigateTo(BuildContext context, Widget screen) {
    Navigator.of(context).push(
      PageRouteBuilder(
        pageBuilder: (context, animation, secondaryAnimation) => screen,
        transitionsBuilder: (context, animation, secondaryAnimation,
child) {
          var begin = Offset(1.0, 0.0);
          var end = Offset.zero;
          var curve = Curves.ease;
```

```
          var tween =
              Tween(begin: begin, end: end).chain(CurveTween(curve:
curve));
          var offsetAnimation = animation.drive(tween);

          return SlideTransition(
            position: offsetAnimation,
            child: child,
          );
        },
      ),
    );
  }

  void _navigateToAllCounselors(BuildContext context) {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) => AllCounselorsScreen(),
      ),
    );
  }
}

class AllCounselorsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          'All Counselors',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
      ),
      body: StreamBuilder(
        stream:
```

```
FirebaseFirestore.instance.collection('counselors').snapshots(),
        builder: (BuildContext context, AsyncSnapshot<QuerySnapshot>
snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          }
          // Display all counselors in a list
          return ListView.builder(
            itemCount: snapshot.data!.docs.length,
            itemBuilder: (context, index) {
              final counselor = snapshot.data!.docs[index];
              return ListTile(
                title: Text(counselor['fullname']),
                subtitle:
                    Text('Price Per Session: ${counselor['price'] ??
'N/A'}'),
                onTap: () {
                  _navigateToCounselorProfile(context, counselor);
                },
              );
            },
          );
        },
      ),
    );
  }

  void _navigateToCounselorProfile(
      BuildContext context, DocumentSnapshot counselor) {
    Navigator.of(context).push(
      MaterialPageRoute(
        builder: (context) => CounselorProfileScreen(counselor:
counselor),
      ),
    );
  }
```

```
}

String? getCurrentUserId() {
  // Get the current user from FirebaseAuth
  User? user = FirebaseAuth.instance.currentUser;

  // Check if a user is signed in
  if (user != null) {
    // User is signed in, return their user ID
    return user.uid;
  } else {
    // No user signed in, return null
    return null;
  }
}

void _addToUserList(BuildContext context, DocumentSnapshot counselor)
async {
  // Get the current user's information from FirebaseAuth
  User? user = FirebaseAuth.instance.currentUser;

  if (user != null) {
    String userId = user.uid;

    // Add the userId of the patient to the counselor's patient list
    try {
      await FirebaseFirestore.instance
          .collection('users')
          .doc(userId)
          .collection('professional_counselors')
          .doc(counselor.id)
          .set({
        'userId': userId,
        'fullname': counselor['fullname'],
        'specialization': counselor['specialization'],
        // Add other patient details as needed
      });

// Also add the patient's userId to the counselor's collection
```

```
      await FirebaseFirestore.instance
          .collection('counselors')
          .doc(counselor.id)
          .collection('patients')
          .doc(userId)
          .set({
        'userId': userId,
        // Add other counselor details as needed
      });
      print('Counselor added to Firestore'); // Add this line

      // Show a snackbar to indicate success
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Counselor added to your list')),
      );

      // Trigger a state update in CounselorScreen to refresh the list
of counselors
      Navigator.pop(context);
    } catch (error) {
      print('Failed to add counselor: $error'); // Add this line

      // Show a snackbar to indicate failure
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Failed to add counselor to your list')),
      );
    }
  }
}

class CounselorProfileScreen extends StatelessWidget {
  final DocumentSnapshot counselor;

  CounselorProfileScreen({required this.counselor});

  @override
  Widget build(BuildContext context) {
    String profilePictureURL = counselor['profileImageUrl'];
    return Scaffold(
```

```
      appBar: AppBar(
        title: Text(
          'Counselor Profile',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            // Display counselor's profile image, name, and other
details
            // You can customize this part according to your UI
requirements
            ClipOval(
              child: Container(
                width: 150, // Adjust the width and height as needed
                height: 150,
                child: CircleAvatar(
                  radius: 50,
                  backgroundImage: NetworkImage(profilePictureURL),
                ),
              ),
            ),
            SizedBox(height: 20),
            Text(
              counselor['fullname'],
              style: TextStyle(
                fontSize: 20.0,
                fontWeight: FontWeight.bold,
              ),
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 10),
```

```
            Text(
              'Age: ${counselor['age'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Education: ${counselor['education'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Email: ${counselor['email'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Experience: ${counselor['experience'] ?? 'N/A'} years',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Location: ${counselor['location'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Price per Session: \$${counselor['price'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            Text(
              'Specialization: ${counselor['specialization'] ?? 'N/A'}',
              style: TextStyle(fontSize: 16.0),
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 20),

            ElevatedButton(
              onPressed: () {
```

```
                _addToUserList(context, counselor);
              },
              child: Text('Add to your list'),
            ),
          ],
        ),
      ),
    );
  }
}

void _removeFromUserList(
    BuildContext context, DocumentSnapshot counselor) async {
  // Get the current user's information from FirebaseAuth
  User? user = FirebaseAuth.instance.currentUser;

  if (user != null) {
    String userId = user.uid;

    // Remove the counselor from the user's list
    try {
      await FirebaseFirestore.instance
          .collection('users')
          .doc(userId)
          .collection('professional_counselors')
          .doc(counselor.id)
          .delete();

      // Show a snackbar to indicate success
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Counselor removed from your list')),
      );

      // Trigger a state update in CounselorScreen to refresh the list
of counselors
      Navigator.pop(context);
    } catch (error) {
      // Show a snackbar to indicate failure
      ScaffoldMessenger.of(context).showSnackBar(
```

```
        SnackBar(content: Text('Failed to remove counselor from your
list')),
      );
    }
  }
}
```

Counselorchatlist_screen.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:safespace/userscreen/counselor_home_screen.dart';
import 'package:safespace/userscreen/counselor_patient_screen.dart';
import 'package:safespace/userscreen/counselor_screen.dart';
import 'package:safespace/userscreen/messagescreen.dart';

class CounselorChatScreen extends StatefulWidget {
  @override
  _CounselorChatScreenState createState() =>
_CounselorChatScreenState();
}

class _CounselorChatScreenState extends State<CounselorChatScreen> {
  int _selectedIndex = 2; // Assuming Chat is the current screen

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: const Text(
          'Chat with Patients',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
```

```
    ),
    body: _buildPatientList(), // Show list of patients
    bottomNavigationBar: BottomNavigationBar(
      items: const <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: 'Profile',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.people),
          label: 'Patients',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.chat),
          label: 'Chat',
        ),
      ],
      currentIndex: _selectedIndex,
      selectedItemColor: Colors.blue,
      onTap: _onItemTapped,
    ),
  );
}

Widget _buildPatientList() {
  return StreamBuilder<QuerySnapshot>(
    stream: FirebaseFirestore.instance
        .collection('counselors')
        .doc(getCurrentUserId()) // Replace with actual counselor ID
        .collection('patients')
        .snapshots(),
    builder: (context, snapshot) {
      if (!snapshot.hasData || snapshot.data == null) {
        return const Center(
          child: CircularProgressIndicator(),
        );
      }
      return ListView.builder(
        itemCount: snapshot.data!.docs.length,
```

```
          itemBuilder: (context, index) {
            var patientId = snapshot.data!.docs[index]
                ['userId']; // Retrieving userId from 'patients'
collection
            return StreamBuilder<DocumentSnapshot>(
              stream: FirebaseFirestore.instance
                  .collection('users')
                  .doc(patientId)
                  .snapshots(),
              builder: (context, userSnapshot) {
                if (!userSnapshot.hasData || userSnapshot.data == null)
{
                  return Container(); // Placeholder until user data is
fetched
                }
                var userData =
                    userSnapshot.data!.data() as Map<String, dynamic>;
                var profileImageUrl = userData['profileImageUrl'];
                var name = userData['name'] ??
                    'Unknown'; // Assuming 'fullname' is the field for
the user's name
                return ListTile(
                  leading: CircleAvatar(
                    backgroundImage: profileImageUrl != null
                        ? NetworkImage(profileImageUrl)
                        : null,
                    child: profileImageUrl == null
                        ? const Icon(Icons.person)
                        : null,
                  ),
                  title: Text(name),
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => MessageScreen(
                          userId: FirebaseAuth.instance.currentUser?.uid
??
                              "", // Use the counselor's ID as the
```

```
userId
                          userType:
                              'counselor', // The userType is
'counselor' because the counselor is the current user
                          recipientId:
                              patientId, // Pass the patientId as the
recipientId
                        ),
                      ),
                    );
                  },
                );
              },
            );
          },
        );
      },
    );
  }

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
    switch (index) {
      case 0:
        // Navigate to profile screen
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) =>
CounselorHomeScreen()),
        );
        break;
      case 1:
        // Navigate to patients screen
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => PatientScreen()),
        );
```

```
        break;
      case 2:
        // No action needed as the user is already on the Chat screen
        break;
    }
  }
}
```

Daily_quote_service.dart

```
class DailyQuoteService {
  static List<String> quotes = [
    "There is hope, even when your brain tells you there isnt.",
    "The true definition of mental illness is when the majority of your time is spent in the past or future, but rarely living in the realism of NOW.",
    "No one would ever say that someone with a broken arm or a broken leg is less than a whole person, but people say that or imply that all the time about people with mental illness.",
    "Its okay not to be okay",
    "You are not your mental illness.",
    "Your struggles do not define you.",
    "Taking care of your mental health is an act of self-love.",
    "You are worthy of happiness and peace of mind."
    // Add more quotes as needed
  ];

  static String getQuoteOfDay(DateTime date) {
    // Use date to generate index for selecting the quote
    int index = date.day % quotes.length;
    return quotes[index];
  }
}
```

Group_chat_screen.dart

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

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

class GroupChatScreen extends StatelessWidget {
  final List<GroupChat> groupChats = [
    GroupChat(id: 1, name: 'Group Chat 1'),
    GroupChat(id: 2, name: 'Group Chat 2'),
    GroupChat(id: 3, name: 'Group Chat 3'),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFF377671),
        title: const Text(
          'Group Chats',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
      ),
      body: ListView.builder(
        itemCount: groupChats.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text(groupChats[index].name),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => GroupChatConversationScreen(
                    groupChat: groupChats[index],
                  ),
                ),
              );
            },
          );
```

```
        },
      ),
    );
  }
}

class GroupChat {
  final int id;
  final String name;

  GroupChat({
    required this.id,
    required this.name,
  });
}

class GroupChatConversationScreen extends StatefulWidget {
  final GroupChat groupChat;

  const GroupChatConversationScreen({Key? key, required this.groupChat})
      : super(key: key);

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

class _GroupChatConversationScreenState
    extends State<GroupChatConversationScreen> {
  final TextEditingController _messageController =
TextEditingController();
  List<ChatMessage> _messages = [];
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  @override
  void initState() {
    super.initState();
    _subscribeToMessages();
  }
```

```
  void _sendMessage() async {
    String messageContent = _messageController.text.trim();
    if (messageContent.isNotEmpty) {
      String senderId = FirebaseAuth.instance.currentUser?.uid ?? "";
      if (senderId.isNotEmpty) {
        DocumentSnapshot userDoc =
            await _firestore.collection('users').doc(senderId).get();
        String senderName =
            (userDoc.data() as Map<String, dynamic>)['fullname'] ??
"Anonymous";
        _firestore
            .collection('group_chats')
            .doc(widget.groupChat.id.toString())
            .collection('messages')
            .add({
          'message': messageContent,
          'senderId':
              FirebaseAuth.instance.currentUser?.uid ?? "", // Add this
line
          'sender': senderName,
          'timestamp': FieldValue.serverTimestamp(),
        });

        _messageController.clear();
      }
    }
  }

  void _subscribeToMessages() {
    String currentUserId = FirebaseAuth.instance.currentUser?.uid ?? "";

    _firestore
        .collection('group_chats')
        .doc(widget.groupChat.id.toString())
        .collection('messages')
        .orderBy('timestamp', descending: true)
        .snapshots()
        .listen((snapshot) async {
```

```
      print('Received snapshot with ${snapshot.docs.length} docs');
      List<ChatMessage> newMessages = [];
      for (var doc in snapshot.docs) {
        print('Doc data: ${doc.data()}');
        String senderId = doc['senderId'];
        DocumentSnapshot userDoc =
            await _firestore.collection('users').doc(senderId).get();
        String senderName =
            (userDoc.data() as Map<String, dynamic>)['fullname'] ??
"Anonymous";
        newMessages.add(ChatMessage(
          message: doc['message'],
          isCurrentUser: senderId == currentUserId,
          sender: senderName,
          timestamp: doc['timestamp'], // Add this line
        ));
      }
      setState(() {
        print('Setting state with ${newMessages.length} messages');
        _messages = newMessages;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.groupChat.name),
      ),
      body: Column(
        children: [
          Expanded(
            child: ListView.builder(
              reverse: true,
              itemCount: _messages.length,
              itemBuilder: (context, index) {
                return ChatBubble(
                  message: _messages[index].message,
```

```
                isCurrentUser: _messages[index].isCurrentUser,
                sender: _messages[index].sender,
              );
            },
          ),
        ),
        Container(
          color: Colors.grey[200],
          padding: const EdgeInsets.symmetric(horizontal: 16,
vertical: 8),
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: _messageController,
                  decoration: InputDecoration(
                    hintText: 'Type a message...',
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(20),
                    ),
                  ),
                ),
              ),
              const SizedBox(width: 8),
              IconButton(
                icon: const Icon(Icons.send),
                onPressed: () {
                  _sendMessage();
                },
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

```
class ChatMessage {
  final String message;
  final bool isCurrentUser;
  final String? sender;
  final Timestamp? timestamp; // Add this line

  ChatMessage({
    required this.message,
    required this.isCurrentUser,
    this.sender,
    this.timestamp, // Add this line
  });
}

class ChatBubble extends StatelessWidget {
  final String message;
  final bool isCurrentUser;
  final String? sender;
  final Timestamp? timestamp; // Add this line

  ChatBubble({
    required this.message,
    required this.isCurrentUser,
    this.sender,
    this.timestamp, // Add this line
  });

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment:
          isCurrentUser ? CrossAxisAlignment.end :
CrossAxisAlignment.start,
      children: [
        if (sender != null && !isCurrentUser)
          Text(sender!,
              style: const TextStyle(fontSize: 12, color: Colors.grey)),
        if (timestamp != null)
          Text(
```

```
            DateFormat('hh:mm a').format(timestamp!.toDate()),
            style: const TextStyle(fontSize: 12, color: Colors.grey),
          ),
        Align(
          alignment:
              isCurrentUser ? Alignment.centerRight :
Alignment.centerLeft,
          child: Padding(
            padding: const EdgeInsets.symmetric(vertical: 4, horizontal:
8),
            child: Container(
              decoration: BoxDecoration(
                color: isCurrentUser ? Colors.blue[200] :
Colors.grey[200],
                borderRadius: BorderRadius.only(
                  topLeft: const Radius.circular(20),
                  topRight: const Radius.circular(20),
                  bottomLeft: isCurrentUser
                      ? const Radius.circular(0)
                      : const Radius.circular(20),
                  bottomRight: isCurrentUser
                      ? const Radius.circular(20)
                      : const Radius.circular(0),
                ),
              ),
              padding: const EdgeInsets.all(12),
              child: Text(
                message,
                style: TextStyle(
                  color: isCurrentUser ? Colors.white : Colors.black,
                ),
              ),
            ),
          ),
        ),
      ],
    );
  }
}
```

Home_screen.dart

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:safespace/userscreen/chatlist_screen.dart';
import 'package:safespace/userscreen/counselor_screen.dart';
import 'package:safespace/userscreen/journal_screen.dart';
import 'package:safespace/userscreen/profile_screen.dart';
import 'package:table_calendar/table_calendar.dart';
import 'package:safespace/userscreen/daily_quote_service.dart';
import
'package:safespace/userscreen/mental_health_article_service.dart';

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        backgroundColor: Color(0xFF377671),
        title: Text(
          'Home',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            _buildPerWeekCalendar(),
            SizedBox(height: 20),
            _buildQuoteOfDay(),
```

```
            SizedBox(height: 20),
            _buildDailyLogs(),
            SizedBox(height: 20),
            _buildDailyArticles(), // Call _buildDailyArticles here
          ],
        ),
      ),
      bottomNavigationBar: BottomNavigationBar(
        backgroundColor: Colors.blue,
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Counselors',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.book),
            label: 'Journal',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        unselectedItemColor: Color(0xFF366763), // Change unselected
item color
        selectedItemColor:
            Color.fromARGB(255, 150, 167, 144), // Change selected item
color
        currentIndex: 0,
        onTap: (index) {
          switch (index) {
```

```
            case 0:
              // No need to navigate, already on home screen
              break;
            case 1:
              _navigateTo(context, CounselorScreen());
              break;
            case 2:
              _navigateTo(context, ChatScreen());
              break;
            case 3:
              _navigateTo(context, JournalScreen());
              break;
            case 4:
              _navigateTo(context, ProfileScreen());
              break;
          }
        },
      ),
    );
  }

  void _navigateTo(BuildContext context, Widget screen) {
    Navigator.of(context).push(
      PageRouteBuilder(
        pageBuilder: (context, animation, secondaryAnimation) => screen,
        transitionsBuilder: (context, animation, secondaryAnimation,
child) {
          var begin = Offset(1.0, 0.0);
          var end = Offset.zero;
          var curve = Curves.ease;
          var tween =
              Tween(begin: begin, end: end).chain(CurveTween(curve:
curve));
          var offsetAnimation = animation.drive(tween);

          return SlideTransition(
            position: offsetAnimation,
            child: child,
          );
```

```
        },
      ),
    );
  }

  Widget _buildPerWeekCalendar() {
    DateTime now = DateTime.now();
    DateTime firstDayOfWeek = now.subtract(Duration(days: now.weekday -
1));
    DateTime lastDayOfWeek = firstDayOfWeek.add(Duration(days: 6));

    return Padding(
      padding: EdgeInsets.all(16.0),
      child: TableCalendar(
        focusedDay: now,
        firstDay: firstDayOfWeek,
        lastDay: lastDayOfWeek,
        // Customize the calendar as needed
        calendarFormat: CalendarFormat.week,
        headerStyle: HeaderStyle(
          titleCentered: true,
        ),
        calendarStyle: CalendarStyle(
          todayDecoration: BoxDecoration(
            color: Colors.blue[200],
            shape: BoxShape.circle,
          ),
          selectedDecoration: BoxDecoration(
            color: Colors.blue[400],
            shape: BoxShape.circle,
          ),
          todayTextStyle: TextStyle(
            fontWeight: FontWeight.bold,
            fontSize: 18.0,
            color: Colors.white,
          ),
        ),
      ),
    );
```

```
}

Widget _buildQuoteOfDay() {
  String quote = DailyQuoteService.getQuoteOfDay(DateTime.now());
  return Card(
    elevation: 4.0,
    child: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          Text(
            'Quote of the Day',
            style: TextStyle(
              fontSize: 20.0,
              fontWeight: FontWeight.bold,
            ),
          ),
          SizedBox(height: 10.0),
          Text(
            quote,
            style: TextStyle(fontSize: 16.0),
          ),
        ],
      ),
    ),
  );
}

Widget _buildDailyLogs() {
  List<String> moods = [
    'Happy',
    'Sad',
    'Angry',
    'Excited',
    'Calm',
    'Stressed',
    'Tired'
  ];
```

```
    return Builder(
      // Add this line
      builder: (BuildContext context) {
        // Add this line
        return Card(
          elevation: 4.0,
          child: Padding(
            padding: EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                Text(
                  'How are you feeling today?',
                  style: TextStyle(
                    fontSize: 20.0,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                SizedBox(height: 10.0),
                Wrap(
                  spacing: 8.0,
                  runSpacing: 8.0,
                  children: moods.map((mood) {
                    return ChoiceChip(
                      label: Text(mood),
                      selected:
                          false, // You can set this based on user
selection
                      onSelected: (isSelected) {
                        // Handle chip selection
                        if (isSelected) {
                          _saveMood(context, mood); // Use context here
                        }
                      },
                    );
                  }).toList(),
                ),
              ],
```

```
          ),
        ),
      );
    }, // Add this line
  ); // Add this line
}

void _saveMood(BuildContext context, String mood) {
  // Add BuildContext parameter
  // Get the current user's ID
  String userId = FirebaseAuth.instance.currentUser!.uid;

  // Get the current date
  DateTime now = DateTime.now();
  String formattedDate = DateFormat('yyyy-MM-dd').format(now);

  // Save the mood to Firestore
  FirebaseFirestore.instance
      .collection('user_moods')
      .doc(userId)
      .collection('moods')
      .doc(formattedDate)
      .set({
    'mood': mood,
    'date': formattedDate,
  }).then((_) {
    ScaffoldMessenger.of(context).showSnackBar(
      // Show SnackBar
      SnackBar(content: Text('Mood saved successfully: $mood')),
    );
  }).catchError((error) {
    print('Failed to save mood: $error');
  });
}

Widget _buildDailyArticles() {
  List<String> articles = MentalHealthArticleService.getArticles();

  return Card(
```

```
      elevation: 4.0,
      child: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            Text(
              'Articles',
              style: TextStyle(
                fontSize: 20.0,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 10.0),
            Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: articles.map((article) {
                return ListTile(
                  title: Text(article),
                  // You can add more elements such as subtitle, onTap,
etc.
                );
              }).toList(),
            ),
          ],
        ),
      ),
    );
  }
}
```

Journal_screen.dart

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

```dart
import 'package:shared_preferences/shared_preferences.dart';
import 'package:safespace/userscreen/chatlist_screen.dart';
import 'package:safespace/userscreen/counselor_screen.dart';
import 'package:safespace/userscreen/home_screen.dart';
import 'package:safespace/userscreen/profile_screen.dart';

class JournalScreen extends StatefulWidget {
  @override
  _JournalScreenState createState() => _JournalScreenState();
}

class _JournalScreenState extends State<JournalScreen> {
  List<JournalEntry> journals = [];
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  late SharedPreferences _prefs;

  @override
  void initState() {
    super.initState();
    _initializePrefs();
  }

  void _initializePrefs() async {
    _prefs = await SharedPreferences.getInstance();
    _fetchJournalEntries();
  }

  Future<void> _fetchJournalEntries() async {
    try {
      if (FirebaseAuth.instance.currentUser != null) {
        String userId = FirebaseAuth.instance.currentUser!.uid;
        var snapshot = await _firestore
            .collection('users')
            .doc(userId)
            .collection('journalEntries')
            .get();
        List<JournalEntry> loadedJournals = [];

        snapshot.docs.forEach((doc) {
```

```
          var data = doc.data();
          var entry = JournalEntry(
            id: doc.id,
            title: data['Journal Entry Title'],
            entry: data['Journal Entry Content'],
            dateTime: (data['Timestamp'] as Timestamp).toDate(),
          );
          loadedJournals.add(entry);
        });

        setState(() {
          journals = loadedJournals;
        });

        _saveJournalsLocally();
      } else {
        _fetchJournalsLocally();
      }
    } catch (e) {
      print('Error fetching journal entries: $e');
      _fetchJournalsLocally();
    }
  }

  Future<void> _saveJournalsLocally() async {
    List<String> serializedJournals =
        journals.map((entry) => jsonEncode(entry.toJson())).toList();

    await _prefs.setStringList('journalEntries', serializedJournals);
  }

  void _fetchJournalsLocally() {
    List<String>? serializedJournals =
_prefs.getStringList('journalEntries');

    if (serializedJournals != null) {
      List<JournalEntry> loadedJournals = serializedJournals
          .map((json) => JournalEntry.fromJson(jsonDecode(json)))
          .toList();
```

```
      setState(() {
        journals = loadedJournals;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: const Text(
          'Journal',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
      ),
      body: journals.isEmpty
          ? const Center(child: Text('No journals added yet'))
          : ListView.builder(
              itemCount: journals.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Row(
                        mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                        children: [
                          Text(journals[index].title),
                          Row(
                            children: [
                              IconButton(
                                icon: const Icon(Icons.edit),
```

```
                                onPressed: () {
                                  _editJournal(context, index);
                                },
                              ),
                              IconButton(
                                icon: const Icon(Icons.delete),
                                onPressed: () {
                                  _removeJournal(context, index);
                                },
                              ),
                            ],
                          ),
                        ],
                      ),
                      Text(
                        'Date:
${DateFormat.yMd().add_jm().format(journals[index].dateTime)}',
                        style:
                            const TextStyle(fontSize: 12, color:
Colors.grey),
                      ),
                    ],
                  ),
                  onTap: () {
                    _viewJournal(context, journals[index]);
                  },
                );
              },
            ),
      floatingActionButton: FloatingActionButton.extended(
        onPressed: () {
          _addJournal(context);
        },
        icon: const Icon(Icons.note_add),
        label: const Text('Add Journal'),
      ),
      bottomNavigationBar: BottomNavigationBar(
        backgroundColor: const Color(0xFF377671),
        items: const <BottomNavigationBarItem>[
```

```
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Counselors',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.book),
            label: 'Journal',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        unselectedItemColor: Color(0xFF366763),
        selectedItemColor: Color.fromARGB(255, 150, 167, 144),
        currentIndex: 3, // Journal icon
        onTap: (index) {
          switch (index) {
            case 0:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => HomeScreen()),
              );
              break;
            case 1:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
CounselorScreen()),
              );
              break;
```

```
            case 2:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => ChatScreen()),
              );
              break;
            case 3:
              // No need to navigate, already on Journal screen
              break;
            case 4:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
ProfileScreen()),
              );
              break;
          }
        },
      ),
    );
  }

  void _addJournal(BuildContext context) async {
    JournalEntry? newJournal = await showDialog<JournalEntry>(
      context: context,
      builder: (BuildContext context) {
        TextEditingController _titleController =
TextEditingController();
        TextEditingController _entryController =
TextEditingController();
        return AlertDialog(
          title: const Text('Add New Journal'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: const InputDecoration(
                  labelText: 'Title',
```

```
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 8),
              TextField(
                controller: _entryController,
                maxLines: null,
                keyboardType: TextInputType.multiline,
                decoration: const InputDecoration(
                  labelText: 'Journal Entry',
                  border: OutlineInputBorder(),
                ),
              ),
            ],
          ),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop(null);
              },
              child: const Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                String titleText = _titleController.text.trim();
                String entryText = _entryController.text.trim();
                if (titleText.isNotEmpty && entryText.isNotEmpty) {
                  DateTime now = DateTime.now();
                  JournalEntry newEntry = JournalEntry(
                    id: '',
                    title: titleText,
                    entry: entryText,
                    dateTime: now,
                  );
                  _saveJournalEntry(newEntry);
                  Navigator.of(context).pop(newEntry);
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(
```

```
                    content:
                        Text('Please enter both a title and a journal
entry'),
                  ));
                }
              },
              child: const Text('Save'),
            ),
          ],
        );
      },
    );

    if (newJournal != null) {
      setState(() {
        journals.add(newJournal);
      });
    }
  }

  void _viewJournal(BuildContext context, JournalEntry journalEntry) {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => JournalDetailScreen(journalEntry:
journalEntry),
      ),
    );
  }

  void _editJournal(BuildContext context, int index) async {
    JournalEntry? editedJournal = await showDialog<JournalEntry>(
      context: context,
      builder: (BuildContext context) {
        TextEditingController _titleController =
            TextEditingController(text: journals[index].title);
        TextEditingController _entryController =
            TextEditingController(text: journals[index].entry);
        return AlertDialog(
```

```
          title: const Text('Edit Journal'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              TextField(
                controller: _titleController,
                decoration: const InputDecoration(
                  labelText: 'Title',
                  border: OutlineInputBorder(),
                ),
              ),
              const SizedBox(height: 8),
              TextField(
                controller: _entryController,
                maxLines: null,
                keyboardType: TextInputType.multiline,
                decoration: const InputDecoration(
                  labelText: 'Journal Entry',
                  border: OutlineInputBorder(),
                ),
              ),
            ],
          ),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop(null);
              },
              child: const Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                String titleText = _titleController.text.trim();
                String entryText = _entryController.text.trim();
                if (titleText.isNotEmpty && entryText.isNotEmpty) {
                  DateTime now = DateTime.now();
                  JournalEntry editedEntry = JournalEntry(
                    id: journals[index].id,
                    title: titleText,
```

```
                    entry: entryText,
                    dateTime: now,
                  );
                  _updateJournalEntry(index, editedEntry);
                  Navigator.of(context).pop(editedEntry);
                } else {
                  ScaffoldMessenger.of(context).showSnackBar(const
SnackBar(
                    content:
                        Text('Please enter both a title and a journal
entry'),
                  ));
                }
              },
              child: const Text('Save'),
            ),
          ],
        );
      },
    );

    if (editedJournal != null) {
      setState(() {
        journals[index] = editedJournal;
      });
    }
  }

  void _removeJournal(BuildContext context, int index) async {
    if (index < 0 || index >= journals.length) {
      log('Invalid index');
      return;
    }

    final confirmed = await showDialog<bool>(
          context: context,
          builder: (BuildContext dialogContext) {
            return AlertDialog(
              title: const Text('Confirm Delete'),
```

```
              content:
                  const Text('Are you sure you want to delete this
journal?'),
              actions: <Widget>[
                TextButton(
                  onPressed: () =>
Navigator.of(dialogContext).pop(false),
                  child: const Text('No'),
                ),
                TextButton(
                  onPressed: () =>
Navigator.of(dialogContext).pop(true),
                  child: const Text('Yes'),
                ),
              ],
            );
          },
        ) ??
        false;

    if (confirmed) {
      try {
        // Remove from Firestore
        await _deleteJournalEntry(journals[index].id);

        // Remove locally
        setState(() {
          journals.removeAt(index);
        });
      } catch (e) {
        log('Error deleting journal: $e');
        _showErrorDialog(context, e.toString());
      }
    }
  }

  void _showErrorDialog(BuildContext context, String errorMessage) {
    showDialog(
      context: context,
```

```
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Error'),
          content: Text('Failed to delete journal entry:
$errorMessage'),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('OK'),
            ),
          ],
        );
      },
    );
  }

  Future<void> _saveJournalEntry(JournalEntry entry) async {
    try {
      String userId = FirebaseAuth.instance.currentUser!.uid;
      await _firestore
          .collection('users')
          .doc(userId)
          .collection('journalEntries')
          .add({
        'Journal Entry Title': entry.title,
        'Journal Entry Content': entry.entry,
        'Timestamp': entry.dateTime,
      });

      print('Journal entry saved successfully.');
    } catch (e) {
      print('Error saving journal entry: $e');
    }
  }

  void _updateJournalEntry(int index, JournalEntry entry) async {
    try {
```

```
      await _firestore
          .collection('users')
          .doc(FirebaseAuth.instance.currentUser!.uid)
          .collection('journalEntries')
          .doc(entry.id)
          .update({
        'Journal Entry Title': entry.title,
        'Journal Entry Content': entry.entry,
        'Timestamp': entry.dateTime,
      });

      log('Journal entry updated successfully.');
    } catch (e) {
      log('Error updating journal entry: $e');
    }
  }

  Future<void> _deleteJournalEntry(String id) async {
    try {
      String userId = FirebaseAuth.instance.currentUser!.uid;
      await _firestore
          .collection('users')
          .doc(userId)
          .collection('journalEntries')
          .doc(id)
          .delete();
      print('Journal entry deleted successfully.');
    } catch (e) {
      print('Error deleting journal entry: $e');
    }
  }
}

class JournalDetailScreen extends StatelessWidget {
  final JournalEntry journalEntry;

  const JournalDetailScreen({Key? key, required this.journalEntry})
      : super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(journalEntry.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Date:
${DateFormat.yMd().add_jm().format(journalEntry.dateTime)}',
              style: const TextStyle(fontWeight: FontWeight.bold),
            ),
            const SizedBox(height: 8),
            Text(journalEntry.entry),
          ],
        ),
      ),
    );
  }
}

class JournalEntry {
  final String id;
  final String title;
  final String entry;
  final DateTime dateTime;

  JournalEntry({
    required this.id,
    required this.title,
    required this.entry,
    required this.dateTime,
  });

  Map<String, dynamic> toJson() {
```

```
    return {
      'id': id,
      'title': title,
      'entry': entry,
      'dateTime': dateTime.toIso8601String(),
    };
  }

  factory JournalEntry.fromJson(Map<String, dynamic> json) {
    return JournalEntry(
      id: json['id'],
      title: json['title'],
      entry: json['entry'],
      dateTime: DateTime.parse(json['dateTime']),
    );
  }
}

void main() {
  runApp(MaterialApp(
    home: JournalScreen(),
  ));
}
```

Mental_health_article_service.dart

```
class MentalHealthArticleService {
  static List<String> articles = [
    "Article 1",
    "Article 2",
    "Article 3",
    // Add more articles as needed
  ];

  static List<String> getArticles() {
    return articles;
  }
}
```

Messagescreen.dart

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

class MessageScreen extends StatefulWidget {
  final String userId;
  final String userType;
  final String recipientId; // Add this field

  MessageScreen(
      {required this.userId,
      required this.userType,
      required this.recipientId}); // Update this line

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

class _MessageScreenState extends State<MessageScreen> {
  TextEditingController _messageController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    Stream<QuerySnapshot> stream = FirebaseFirestore.instance
        .collection('messages')
        .orderBy('timestamp')
        .snapshots();

    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Chat',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
```

```
        ),
      ),
      body: Column(
        children: [
          Expanded(
              child: StreamBuilder(
            stream: stream,
            builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {
              if (snapshot.connectionState == ConnectionState.waiting) {
                return const Center(child: CircularProgressIndicator());
              }
              if (snapshot.hasError) {
                return Center(child: Text('Error: ${snapshot.error}'));
              }

              List<QueryDocumentSnapshot> docs = snapshot.data!.docs;
              docs = docs.where((doc) {
                bool isSender = doc['sender'] == widget.userId ||
                    doc['sender'] == widget.recipientId;
                bool isRecipient = doc['recipient'] == widget.userId ||
                    doc['recipient'] == widget.recipientId;
                return isSender && isRecipient;
              }).toList();
              docs.sort((a, b) => (b['timestamp'] as Timestamp)
                  .compareTo(a['timestamp'] as Timestamp));

              return ListView.builder(
                itemCount: docs.length,
                reverse: true,
                itemBuilder: (context, index) {
                  var message = docs[index];
                  bool isMe = message['sender'] ==
                      FirebaseAuth.instance.currentUser?.uid;
                  return Container(
                    alignment:
                        isMe ? Alignment.centerRight :
Alignment.centerLeft,
                    child: Container(
                      margin: const EdgeInsets.all(8.0),
```

```
                    padding: const EdgeInsets.all(10.0),
                    decoration: BoxDecoration(
                      color: isMe ? Colors.blue[200] :
Colors.grey[200],
                      borderRadius: BorderRadius.circular(10.0),
                    ),
                    child: Text(message['content']),
                  ),
                );
              },
            );
          },
        )),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Row(
            children: [
              Expanded(
                child: TextField(
                  controller: _messageController,
                  decoration: InputDecoration(
                    hintText: 'Type a message...',
                    filled: true,
                    fillColor: Colors.white,
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(30.0),
                      borderSide: BorderSide.none,
                    ),
                    contentPadding: EdgeInsets.symmetric(
                        horizontal: 20.0, vertical: 10.0),
                    suffixIcon: IconButton(
                      icon: const Icon(Icons.send),
                      onPressed: () {
                        _sendMessage();
                      },
                    ),
                  ),
                ),
              ),
```

```
              ],
            ),
          ),
        ],
      ),
    );
  }

  void _sendMessage() {
    String messageContent = _messageController.text.trim();
    if (messageContent.isNotEmpty) {
      String senderId = widget.userId; // The senderId is the userId
      String recipientId =
          widget.recipientId; // The recipientId is the recipientId

      if (senderId.isNotEmpty && recipientId.isNotEmpty) {
        FirebaseFirestore.instance.collection('messages').add({
          'sender': senderId,
          'recipient': recipientId,
          'content': messageContent,
          'timestamp': DateTime.now(),
        }).then((_) {
          _messageController.clear();
        }).catchError((error) {
          print("Error sending message: $error");
        });
      } else {
        print("Sender or recipient ID is empty.");
      }
    }
  }
}
```

Patientdetailscreen.dart

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

class PatientDetailsScreen extends StatelessWidget {
```

```dart
  final String patientUserId;

  PatientDetailsScreen(this.patientUserId);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        automaticallyImplyLeading: false,
        title: Text(
          'Patient Details',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
      ),
      body: FutureBuilder(
        future: FirebaseFirestore.instance
            .collection('users')
            .doc(patientUserId)
            .get(),
        builder: (context, AsyncSnapshot<DocumentSnapshot> userSnapshot)
{
          if (userSnapshot.connectionState == ConnectionState.waiting) {
            return Center(child: CircularProgressIndicator());
          }
          if (userSnapshot.hasError) {
            return Center(child: Text('Error: ${userSnapshot.error}'));
          }
          if (!userSnapshot.hasData || userSnapshot.data == null) {
            return Center(child: Text('User not found'));
          }
          final userData =
              userSnapshot.data?.data() as Map<String, dynamic>? ?? {};
          final fullname = userData['fullname'] as String? ?? 'N/A';
          final gender = userData['gender'] as String? ?? 'N/A';
          final email = userData['email'] as String? ?? 'N/A';
```

```
          final interest = userData['interest'] as String? ?? 'N/A';
          final location = userData['location'] as String? ?? 'N/A';
          final profilePictureURL = userData['profilePictureURL'] as
String?;

          return Padding(
            padding: EdgeInsets.all(16.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                if (profilePictureURL != null)
                  CircleAvatar(
                    backgroundImage: NetworkImage(profilePictureURL),
                    radius: 50,
                  ),
                SizedBox(height: 16),
                Text('Fullname: $fullname'),
                Text('Gender: $gender'),
                Text('Email: $email'),
                Text('Interest: $interest'),
                Text('Location: $location'),
              ],
            ),
          );
        },
      ),
    );
  }
}
```

Profile_screen.dart

```
import 'dart:io';

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

```dart
import 'package:safespace/userscreen/chatlist_screen.dart';
import 'package:safespace/userscreen/counselor_screen.dart';
import 'package:safespace/userscreen/home_screen.dart';
import 'package:safespace/userscreen/journal_screen.dart';
import 'package:safespace/userscreen/signin_screen.dart';
import 'package:safespace/userscreen/user_changepassword.dart';

class ProfileScreen extends StatefulWidget {
  @override
  _ProfileScreenState createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
  late User _currentUser;
  String? _username;
  late String _profilePictureURL = '';

  @override
  void initState() {
    super.initState();
    _currentUser = FirebaseAuth.instance.currentUser!;
    _fetchUsername();
  }

  Future<void> _fetchUsername() async {
    try {
      final DocumentSnapshot userDoc = await FirebaseFirestore.instance
          .collection('users')
          .doc(_currentUser.uid)
          .get();
      setState(() {
        _username = userDoc['fullname'];
        _profilePictureURL = userDoc['profilePictureURL'];
      });
    } catch (e) {
      print('Error fetching username: $e');
    }
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color(0xFF377671),
        automaticallyImplyLeading: false, // Remove back button
        title: Text(
          'Profile',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.white,
            fontSize: 25,
          ),
        ),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              CircleAvatar(
                  radius: 50,
                  backgroundImage: NetworkImage(_profilePictureURL)),
              SizedBox(height: 20),
              Text(
                _username ?? 'Username',
                style: TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
              ),
              Text(
                _currentUser.email ??
                    'user@example.com', // Replace with actual email
                style: TextStyle(fontSize: 16),
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
```

```
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return ViewProfileDialog(user: _currentUser);
                  },
                );
              },
              child: Text('View Profile'),
            ),
            SizedBox(height: 20),
            ListTile(
              leading: Icon(Icons.lock),
              title: Text('Change Password'),
              onTap: () {
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return ChangePasswordDialog(user: _currentUser);
                  },
                );
              },
            ),
            ListTile(
              leading: Icon(Icons.policy),
              title: Text('Terms of Service'),
              onTap: () {
                // Show Terms of Service dialog
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return TermsOfServiceDialog();
                  },
                );
              },
            ),
            ListTile(
              leading: Icon(Icons.phone),
              title: Text('Emergency Hotline'),
              onTap: () {
```

```
                // Show emergency hotline dialog
                showDialog(
                  context: context,
                  builder: (BuildContext context) {
                    return EmergencyHotlineDialog();
                  },
                );
              },
            ),
            ListTile(
              leading: Icon(Icons.logout),
              title: Text('Logout'),
              onTap: () async {
                try {
                  await FirebaseAuth.instance.signOut(); // Sign out
the user
                  Navigator.pushReplacement(
                    context,
                    MaterialPageRoute(
                        builder: (context) => LoginRegisterScreen()),
                  );
                } catch (e) {
                  print('Error signing out: $e');
                  // Handle sign-out error
                }
              },
            ),
          ],
        ),
      ),
    ),
    bottomNavigationBar: BottomNavigationBar(
      backgroundColor: const Color(0xFF377671),
      items: const <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.home),
          label: 'Home',
        ),
        BottomNavigationBarItem(
```

```
            icon: Icon(Icons.people),
            label: 'Counselors',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.chat),
            label: 'Chat',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.book),
            label: 'Journal',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
        ],
        unselectedItemColor: Color(0xFF366763),
        selectedItemColor: Color.fromARGB(255, 150, 167, 144),
        currentIndex: 4,
        onTap: (index) {
          switch (index) {
            case 0:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => HomeScreen()),
              );
              break;
            case 1:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
CounselorScreen()),
              );
              break;
            case 2:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) => ChatScreen()),
              );
```

```
              break;
            case 3:
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
JournalScreen()),
              );
              break;
            case 4:
              // No need to navigate, already on profile screen
              break;
          }
        },
      ),
    );
  }
}

class ViewProfileDialog extends StatefulWidget {
  final User user;

  ViewProfileDialog({required this.user});

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

class _ViewProfileDialogState extends State<ViewProfileDialog> {
  late String _name = '';
  late String _gender = '';
  late String _interest = '';
  late String _location = '';
  late String _username = '';
  late String _profilePictureURL = '';

  @override
  void initState() {
    super.initState();
    _fetchUserProfile();
```

```
}

Future<void> _fetchUserProfile() async {
  try {
    final DocumentSnapshot userDoc = await FirebaseFirestore.instance
        .collection('users')
        .doc(widget.user.uid)
        .get();
    setState(() {
      _name = userDoc['name'];
      _gender = userDoc['gender'];
      _interest = userDoc['interest'];
      _location = userDoc['location'];
      _username = userDoc['fullname'];
      _profilePictureURL = userDoc['profilePictureURL'];
    });
  } catch (e) {
    print('Error fetching user profile: $e');
  }
}

@override
Widget build(BuildContext context) {
  return AlertDialog(
    title: Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        Text(
          'View Profile',
          style: TextStyle(
            fontFamily: 'poetsen',
            color: Colors.black,
            fontSize: 25,
          ),
        ),
        IconButton(
          icon: Icon(Icons.edit),
          onPressed: () {
            Navigator.pop(context);
```

```
              showDialog(
                context: context,
                builder: (BuildContext context) {
                  return EditProfileDialog(user: widget.user);
                },
              );
            },
          ),
        ],
      ),
      content: SingleChildScrollView(
        child: Column(
          mainAxisSize: MainAxisSize.min,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            CircleAvatar(
                radius: 50, backgroundImage:
NetworkImage(_profilePictureURL)),
            SizedBox(height: 20),
            Text(
              'Username: $_username',
              style: TextStyle(fontWeight: FontWeight.bold),
            ),
            Text('Email: ${widget.user.email ?? 'N/A'}'),
            SizedBox(height: 10),
            Text('Name: $_name'),
            Text('Gender: $_gender'),
            Text('Interest: $_interest'),
            Text('Location: $_location'),
          ],
        ),
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Close'),
        ),
```

```
      ],
    );
  }
}

class EditProfileDialog extends StatefulWidget {
  final User user;

  EditProfileDialog({required this.user});

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

class _EditProfileDialogState extends State<EditProfileDialog> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  late String _name;
  late String _gender;
  late String _interest;
  late String _location;
  late String _profilePictureURL;

  @override
  void initState() {
    super.initState();
    _name = widget.user.displayName ?? '';
    _gender = '';
    _interest = '';
    _location = '';
    _profilePictureURL = ''; // Initialize with empty string
  }

  Future<void> _uploadProfilePicture(File imageFile) async {
    try {
      // Upload the image file to Firebase Storage
      String imageName =
          'profile_picture_${DateTime.now().millisecondsSinceEpoch}';
      Reference storageReference =
```

```
FirebaseStorage.instance.ref().child('profile_pictures/$imageName');
      UploadTask uploadTask = storageReference.putFile(imageFile);
      TaskSnapshot taskSnapshot = await uploadTask;
      String downloadURL = await taskSnapshot.ref.getDownloadURL();

      // Update the profile picture URL in the state
      setState(() {
        _profilePictureURL = downloadURL;
      });
    } catch (e) {
      print('Error uploading profile picture: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Edit Profile'),
      content: SingleChildScrollView(
        child: Form(
          key: _formKey,
          child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
              // Profile Picture Upload
              ElevatedButton(
                onPressed: () async {
                  // Open image picker
                  final ImagePicker _picker = ImagePicker();
                  final pickedFile =
                      await _picker.pickImage(source:
ImageSource.gallery);
                  if (pickedFile != null) {
                    File imageFile = File(pickedFile.path);
                    await _uploadProfilePicture(imageFile);
                  }
                },
                child: Text('Upload Profile Picture'),
              ),
```

```
            // Display uploaded profile picture if available
            if (_profilePictureURL.isNotEmpty)
              Image.network(
                _profilePictureURL,
                height: 100,
              ),
            TextFormField(
              initialValue: _name,
              decoration: InputDecoration(labelText: 'Name'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your name';
                }
                return null;
              },
              onSaved: (value) => _name = value!,
            ),
            TextFormField(
              decoration: InputDecoration(labelText: 'Gender'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your gender';
                }
                return null;
              },
              onSaved: (value) => _gender = value!,
            ),
            TextFormField(
              decoration: InputDecoration(labelText: 'Interest'),
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your interest';
                }
                return null;
              },
              onSaved: (value) => _interest = value!,
            ),
            TextFormField(
              decoration: InputDecoration(labelText: 'Location'),
```

```
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your location';
                }
                return null;
              },
              onSaved: (value) => _location = value!,
            ),
          ],
        ),
      ),
    ),
    actions: [
      TextButton(
        onPressed: () {
          Navigator.pop(context);
        },
        child: Text('Cancel'),
      ),
      ElevatedButton(
        onPressed: () async {
          if (_formKey.currentState!.validate()) {
            _formKey.currentState!.save();
            try {
              await FirebaseFirestore.instance
                  .collection('users')
                  .doc(widget.user.uid)
                  .update({
                'name': _name,
                'gender': _gender,
                'interest': _interest,
                'location': _location,
                'profilePictureURL':
                    _profilePictureURL, // Update profile picture URL
              });
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text('Profile updated successfully'),
                  duration: Duration(seconds: 2),
```

```
                ),
              );
              Navigator.pop(context);
            } catch (e) {
              print('Error updating profile: $e');
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content:
                      Text('Failed to update profile. Please try
again.'),
                  duration: Duration(seconds: 2),
                ),
              );
            }
          }
        },
        child: Text('Save'),
      ),
    ],
  );
 }
}

class TermsOfServiceDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Terms of Service'),
      content: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              '1. Acceptance of Terms: By using this app, you agree to
abide by these Terms of Service.',
            ),
            Text(
              '2. User Conduct: Users must not engage in any activity
that disrupts the app\'s functionality or violates the rights of
```

```
others.',
            ),
            Text(
              '3. Privacy Policy: Users should also review the app\'s
Privacy Policy, which outlines how their personal information is
collected, used, and protected.',
            ),
            // Add more terms as needed
          ],
        ),
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Close'),
        ),
      ],
    );
  }
}

class EmergencyHotlineDialog extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: Text('Emergency Hotline'),
      content: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'In case of emergency, please contact the following
hotlines:',
              style: TextStyle(fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 10),
            Text('National Emergency Hotline: 911'),
```

```
            Text('Police: 117'),
            Text('Fire Department: 118'),
            Text('Medical Emergency: 112'),
          ],
        ),
      ),
      actions: [
        TextButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Close'),
        ),
      ],
    );
  }
}

void main() {
  runApp(MaterialApp(
    home: ProfileScreen(),
  ));
}
```

Signin_screen.dart

```
import 'package:flutter/material.dart';
import 'package:safespace/userscreen/counselor_home_screen.dart'; //
Import the counselor home screen
import 'package:safespace/userscreen/home_screen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

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

  @override
  _LoginRegisterScreenState createState() =>
_LoginRegisterScreenState();
```

```dart
}

class _LoginRegisterScreenState extends State<LoginRegisterScreen>
    with SingleTickerProviderStateMixin {
  late TabController _tabController;

  @override
  void initState() {
    super.initState();
    _tabController = TabController(length: 2, vsync: this);
  }

  Future<void> _signIn(
      String usernameOrEmail, String password, UserType userType) async
{
    try {
      UserCredential userCredential = await FirebaseAuth.instance
          .signInWithEmailAndPassword(
              email: usernameOrEmail, password: password);

      // Check if the signed-in user is authorized for the selected tab
      if ((userType == UserType.User &&
              userCredential.user!.email!.contains('@user.com')) ||
          (userType == UserType.Counselor &&
              userCredential.user!.email!.contains('@counselor.com'))) {
        // Sign in successful, navigate to HomeScreen
        if (userType == UserType.Counselor) {
          // Navigate to CounselorHomeScreen
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) =>
CounselorHomeScreen()),
          );
        } else {
          // Navigate to HomeScreen for other users
          Navigator.pushReplacement(
            context,
            MaterialPageRoute(builder: (context) => HomeScreen()),
          );
```

```
        }
      } else {
        // Not authorized, show error dialog
        showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: const Text('Login Failed'),
              content: const Text('You are not authorized to access this
tab.'),
              actions: [
                TextButton(
                  onPressed: () {
                    Navigator.pop(context);
                  },
                  child: const Text('OK'),
                ),
              ],
            );
          },
        );
      }
    } catch (e) {
      // Sign in failed, show error dialog
      showDialog(
        context: context,
        builder: (context) {
          return AlertDialog(
            title: Text('Login Failed'),
            content: Text('Invalid email or password. Please try
again.'),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: Text('OK'),
              ),
            ],
```

```
          );
        },
      );
    }
  }

  Future<void> _register(String usernameOrEmail, String email, String
password,
      UserType userType) async {
    showDialog(
      context: context,
      builder: (context) {
        return RegisterDialog(
          onRegister: (fullname, email, password) async {
            try {
              UserCredential userCredential = await
FirebaseAuth.instance
                  .createUserWithEmailAndPassword(
                      email: email, password: password);
              print('User registered: ${userCredential.user!.uid}');

              // Get the user's unique ID
              String userId = userCredential.user!.uid;

              // Save user information to Firestore
              await FirebaseFirestore.instance
                  .collection(
                      userType == UserType.User ? 'users' :
'counselors')
                  .doc(userId)
                  .set({
                'fullname': fullname,
                'email': email,
              });

              // Print user registration success message
              print('User registered: $userId');

              // Navigate to HomeScreen or perform other actions as
```

```
needed
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => HomeScreen()),
              );
            } catch (e) {
              // Handle registration errors
              print('Error registering user: $e');
            }
          },
          isCounselor: userType == UserType.Counselor,
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color(0xFF377671),
        automaticallyImplyLeading: false,
        title: Row(
          children: <Widget>[
            Image.asset('lib/assets/logo.png', fit: BoxFit.contain,
height: 80),
            SizedBox(width: 8), // You can adjust the size as per your
need
            Text(
              'Welcome to SafeSpace',
              style: TextStyle(
                fontFamily: 'poetsen',
                color: Colors.white,
                fontSize: 25,
              ),
            ),
          ],
        ),
      ),
```

```
      body: Column(
        children: [
          TabBar(
            controller: _tabController,
            labelColor: Color(0xFF11736C),
            tabs: [
              Tab(text: 'USER'),
              Tab(text: 'COUNSELOR'),
            ],
            indicatorColor: Color(0xFF11736C), // Change TabBar
indicator color
          ),
          Expanded(
            child: TabBarView(
              controller: _tabController,
              children: [
                SingleChildScrollView(
                  // Add this line
                  child: LoginRegisterTab(
                    onSignIn: _signIn,
                    onRegister: _register,
                    userType: UserType.User,
                  ),
                ),
                SingleChildScrollView(
                  // Add this line
                  child: LoginRegisterTab(
                    onSignIn: _signIn,
                    onRegister: _register,
                    userType: UserType.Counselor,
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
```

```
  @override
  void dispose() {
    _tabController.dispose();
    super.dispose();
  }
}

class LoginRegisterTab extends StatefulWidget {
  final Function(String usernameOrEmail, String password, UserType
userType)
      onSignIn;
  final Function(
          String username, String email, String password, UserType
userType)
      onRegister;
  final UserType userType;

  const LoginRegisterTab({
    required this.onSignIn,
    required this.onRegister,
    required this.userType,
  });

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

class _LoginRegisterTabState extends State<LoginRegisterTab> {
  late TextEditingController _usernameOrEmailController;
  late TextEditingController _passwordController;

  @override
  void initState() {
    super.initState();
    _usernameOrEmailController = TextEditingController();
    _passwordController = TextEditingController();
  }
```

```
  @override
  void dispose() {
    _usernameOrEmailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  void _showCounselorRegisterDialog() {
    showDialog(
      context: context,
      builder: (context) {
        return RegisterDialog(
          onRegister: (fullname, email, password) async {
            try {
              UserCredential userCredential = await
FirebaseAuth.instance
                  .createUserWithEmailAndPassword(
                      email: email, password: password);

              // Get the user's unique ID
              String userId = userCredential.user!.uid;

              // Save counselor information to Firestore
              await FirebaseFirestore.instance
                  .collection('counselors')
                  .doc(userId)
                  .set({
                'fullname': fullname,
                'email': email,
                // Add other fields as needed
              });

              // Print counselor registration success message
              print('Counselor registered: $userId');

              // Close the dialog
              Navigator.pop(context);

              // Navigate to HomeScreen or perform other actions as
```

```
needed
              Navigator.pushReplacement(
                context,
                MaterialPageRoute(builder: (context) => HomeScreen()),
              );
            } catch (e) {
              // Handle registration errors
              print('Error registering counselor: $e');
            }
          },
          isCounselor: true, // Always show counselor registration form
        );
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Image.asset('lib/assets/user_login.png'),
          Text(
            '${widget.userType == UserType.User ? 'User' :
'Counselor'}',
            style: TextStyle(
              color: Color(0xFF6E6E6E), // Change color here
            ),
          ),
          Padding(
            padding: EdgeInsets.all(8.0),
            child: TextFormField(
              controller: _usernameOrEmailController,
              decoration: InputDecoration(
                labelText: 'Email',
                hintStyle: TextStyle(color: Color(0xFF6E6E6E)),
                fillColor: Color(0xFFD1D1D1),
```

```
            filled: true,
            border: OutlineInputBorder(
              borderSide: BorderSide.none,
              borderRadius: BorderRadius.circular(10.0),
            ),
          ),
        ),
      ),
      Padding(
        padding: EdgeInsets.all(8.0),
        child: TextFormField(
          controller: _passwordController,
          obscureText: true,
          decoration: InputDecoration(
            labelText: 'Password',
            hintStyle: TextStyle(color: Color(0xFF6E6E6E)),
            fillColor: Color(0xFFD1D1D1),
            filled: true,
            border: OutlineInputBorder(
              borderSide: BorderSide.none,
              borderRadius: BorderRadius.circular(10.0),
            ),
          ),
        ),
      ),
      Padding(
        padding: EdgeInsets.all(8.0),
        child: ElevatedButton(
          onPressed: () {
            widget.onSignIn(
              _usernameOrEmailController.text.trim(),
              _passwordController.text.trim(),
              widget.userType,
            );
          },
          style: ElevatedButton.styleFrom(
            foregroundColor: Color(0xFFEBFCE4),
            backgroundColor: Color(0xFF377671),
          ),
```

```
              child: Text('Login'),
            ),
          ),
          if (widget.userType == UserType.User)
            Padding(
              padding: EdgeInsets.all(8.0),
              child: ElevatedButton(
                onPressed: () {
                  widget.onRegister(
                    _usernameOrEmailController.text.trim(),
                    '', // Empty email for now
                    _passwordController.text.trim(),
                    widget.userType,
                  );
                },
                style: ElevatedButton.styleFrom(
                  foregroundColor: Color(0xFFEBFCE4),
                  backgroundColor: Color(0xFF377671),
                ),
                child: Text('Register'),
              ),
            ),
          if (widget.userType == UserType.Counselor)
            Padding(
              padding: EdgeInsets.all(8.0),
              child: ElevatedButton(
                onPressed: _showCounselorRegisterDialog,
                style: ElevatedButton.styleFrom(
                  foregroundColor: Color(0xFFEBFCE4),
                  backgroundColor: Color(0xFF377671),
                ),
                child: Text('Register as Counselor'),
              ),
            ),
        ],
      ),
    );
  }
}
```

```dart
class RegisterDialog extends StatefulWidget {
  final Function(String fullname, String email, String password)
onRegister;
  final bool isCounselor;

  const RegisterDialog({
    required this.onRegister,
    this.isCounselor = false,
  });

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

class _RegisterDialogState extends State<RegisterDialog> {
  late TextEditingController _usernameOrFullNameController;
  late TextEditingController _emailController;
  late TextEditingController _passwordController;
  late TextEditingController _confirmPasswordController;
  final _formKey = GlobalKey<FormState>(); // Add form key

  @override
  void initState() {
    super.initState();
    _usernameOrFullNameController = TextEditingController();
    _emailController = TextEditingController();
    _passwordController = TextEditingController();
    _confirmPasswordController = TextEditingController();
  }

  @override
  void dispose() {
    _usernameOrFullNameController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    _confirmPasswordController.dispose();
    super.dispose();
  }
```

```
  @override
  Widget build(BuildContext context) {
    String emailPlaceholder = widget.isCounselor
        ? 'examplecounselor@counselor.com'
        : 'exampleuser@user.com';
    String requiredEmailDomain =
        widget.isCounselor ? '@counselor.com' : '@user.com';

    return AlertDialog(
      title: Text(widget.isCounselor ? 'Register as Counselor' :
'Register'),
      content: Form(
        key: _formKey, // Add form key to the Form widget
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            if (!widget.isCounselor)
              TextFormField(
                controller: _usernameOrFullNameController,
                decoration: InputDecoration(
                  labelText: 'Username',
                ),
                validator: (value) {
                  // Validate username format
                  if (value!.length < 6 || value.length > 15) {
                    return 'Username must be between 6 and 15
characters';
                  }
                  if
(!RegExp(r'^(?=.*?[a-zA-Z0-9]).{6,15}$').hasMatch(value)) {
                    return 'Username must contain at least 1 symbol or
number';
                  }
                  return null;
                },
              )
            else
              TextFormField(
```

```
            controller: _usernameOrFullNameController,
            decoration: InputDecoration(
              labelText: 'Fullname (Lastname, Firstname MI)',
            ),
            validator: (value) {
              // Validate full name format
              if (!RegExp(r'^[A-Za-z]+, [A-Za-z]+(?: [A-Za-z]\.)?$')
                  .hasMatch(value!)) {
                return 'Invalid full name format';
              }
              return null;
            },
          ),
          SizedBox(height: 10),
          TextFormField(
            controller: _emailController,
            decoration: InputDecoration(
              labelText: 'Email',
              hintText: emailPlaceholder,
            ),
            validator: (value) {
              // Validate email format
              if (!value!.endsWith(requiredEmailDomain)) {
                return 'Email must end with $requiredEmailDomain';
              }
              return null;
            },
          ),
          SizedBox(height: 10),
          TextFormField(
            controller: _passwordController,
            obscureText: true,
            decoration: InputDecoration(
              labelText: 'Password',
            ),
            validator: (value) {
              // Validate password format
              if (!RegExp(
```

```
r'^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[^\w\s]).{8,}$')
                    .hasMatch(value!)) {
                  return 'Password must be at least 8 characters and
contain at least 1 symbol, number, capital letter, and small letter';
                }
                return null;
              },
            ),
            SizedBox(height: 10),
            TextFormField(
              controller: _confirmPasswordController,
              obscureText: true,
              decoration: InputDecoration(
                labelText: 'Confirm Password',
              ),
              validator: (value) {
                // Validate password confirmation
                if (value != _passwordController.text) {
                  return 'Passwords do not match';
                }
                return null;
              },
            ),
          ],
        ),
      ),
      actions: [
        ElevatedButton(
          onPressed: () {
            if (_formKey.currentState!.validate()) {
              // Only register if form is valid
              if (_passwordController.text ==
_confirmPasswordController.text) {
                widget.onRegister(
                  _usernameOrFullNameController.text.trim(),
                  _emailController.text.trim(),
                  _passwordController.text.trim(),
                );
                Navigator.pop(context);
```

```
            } else {
              showDialog(
                context: context,
                builder: (context) {
                  return AlertDialog(
                    title: Text('Error'),
                    content: SingleChildScrollView(
                      child: Text(
                        'Passwords do not match. Please make sure your
password contains at least 8 characters with at least 1 symbol, number,
capital letter, and small letter.',
                      ),
                    ),
                    actions: [
                      TextButton(
                        onPressed: () {
                          Navigator.pop(context);
                        },
                        child: Text('OK'),
                      ),
                    ],
                  );
                },
              );
            }
          }
        },
        style: ElevatedButton.styleFrom(
          foregroundColor: Color(0xFFEBFCE4),
          backgroundColor: Color(0xFF377671),
        ),
        child: Text('Register'),
      ),
      TextButton(
        onPressed: () {
          Navigator.pop(context);
        },
        child: Text('Cancel'),
      ),
```

```
      ],
    );
  }
}

enum UserType {
  User,
  Counselor,
}

void main() {
  runApp(MaterialApp(
    home: LoginRegisterScreen(),
  ));
}
```

Splash_screen.dart

```
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:safespace/userscreen/signin_screen.dart';

class SplashScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Simulate loading time with a delay
    Timer(Duration(seconds: 2), () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => LoginRegisterScreen()),
      );
    });

    return Scaffold(
      body: Center(
        child: CircularProgressIndicator(), // Or any other loading
indicator
      ),
```

```
    );
  }
}
```

Table_calendar.dart

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

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

  @override
  // ignore: library_private_types_in_public_api
  _TableCalendarWidgetState createState() =>
_TableCalendarWidgetState();
}

class _TableCalendarWidgetState extends State<TableCalendarWidget> {
  CalendarFormat _calendarFormat = CalendarFormat.month;
  DateTime _focusedDay = DateTime.now();
  DateTime? _selectedDay;

  @override
  Widget build(BuildContext context) {
    return TableCalendar(
      calendarFormat: _calendarFormat,
      focusedDay: _focusedDay,
      firstDay: DateTime.utc(2020, 1, 1),
      lastDay: DateTime.utc(2030, 12, 31),
      selectedDayPredicate: (day) {
        // Use `selectedDayPredicate` to determine which day is
currently selected.
        // If this returns true, then `day` will be marked as selected.
        return isSameDay(_selectedDay, day);
      },
      onDaySelected: (selectedDay, focusedDay) {
        setState(() {
          _selectedDay = selectedDay;
```

```
          _focusedDay = focusedDay; // Update `_focusedDay` here as well
        });
      },
      onFormatChanged: (format) {
        if (_calendarFormat != format) {
          setState(() {
            _calendarFormat = format;
          });
        }
      },
      onPageChanged: (focusedDay) {
        // No need to call `setState()` here because this page change is
just for visual purpose.
        _focusedDay = focusedDay;
      },
    );
  }
}
```

III. Group Member's Summary of Assigned Task

Name (LN, FN, MI)	Picture	Detailed Contribution/Assigned Tasks
Leader: **Marasigan, Ma. Clarissa C.**		**Program Code** **Scope and Limitation** **Hardware/ Software Specification**

Members: **Limson, Clark Czedrick F.**		**Scope and Limitation** **Hardware/ Software** **Specification**
Lituania, Andrea A.		**Screenshots**