JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

EXPENSE MANAGER

Dr.Prakash S, Nagappan L, Ragul M ,Sri Ganesh S

Head of the Department ,Student ,Student ,Student

Information Technology

Sri Shakthi Institute of Engineering and Technology,Coimbatore,India

Abstract : Expense Manager is a comprehensive mobile application created to make it easier and more efficient for groups of people to manage their shared spending. The issues that friends, relatives, roommates, coworkers, or any group of individuals that split costs on various occasions, such as vacations, household bills, or social gatherings, confront are covered in this project. The user-friendly platform provided by Expense Manager allows users to create and manage many rooms, each of which represents a different expense-sharing group. Users may easily track, record, and settle spending within each room, assuring fairness and effective financial management.

Introduction:

The issue of managing shared spending is all too familiar in a society where shared experiences are more prevalent, whether it be vacations with friends, household bills with roommates, or social occasions with coworkers. Confusion, mistakes, and even disagreements among members within these groups can result from the usual headache of keeping track of who owes what, accurately estimating expenses, and maintaining financial fairness.

Enter Expense Manager, a feature-rich smartphone software created solely to streamline and simplify the management of shared spending among various groups of people. Expense Manager is the answer to these typical problems whether it's a close-knit group of friends, a family, roommates sharing a home, or coworkers working together on social occasions.

Expense Manager doesn't just provide a tool; it offers a user-friendly platform where users can create and manage multiple rooms, each representing a specific expense-sharing group. Within each room, users can seamlessly track, record, and settle expenses. The result is a level of transparency, fairness, and efficient financial management that greatly improves the dynamics within these groups.

In this era of rapidly evolving technology, Expense Manager stands out as a beacon of simplicity and practicality, addressing the universal problem of shared expenses in a manner that eases financial harmony and reduces conflicts. By providing accurate expense calculations, transparent record-keeping, and efficient settlement, Expense Manager becomes the missing link that transforms shared expense management from a chore into a hassle-free and harmonious experience.

Literature Survey:

Introduction to Expense Management and Shared Financial Tracking:

- Define the concept of expense management within groups or collaborative settings.
- Explain the importance of transparent and efficient expense tracking for various contexts.

Existing Expense Management Tools:

- Survey existing software solutions and mobile applications designed for shared expense management.
- Evaluate the strengths and limitations of popular tools in the market.

Collaborative Finance Applications:

- Explore collaborative applications and platforms that enable users to collaborate on financial tasks and track shared expenses.
- Analyze their features, user interfaces, and user experiences.

Group Expense Tracking Systems:

- Review systems designed specifically for tracking shared expenses within groups, including families, roommates, and social circles.
- Discuss the functionalities they offer and their user adoption.

Financial Transparency and Conflict Resolution:

- Discuss the importance of financial transparency and how efficient expense tracking systems contribute to conflict resolution among group members.

Flutter:

- Provides a comprehensive reference for Flutter, including topics like widget building, state management, and UI design.
- "Flutter Design Patterns and Best Practices: Build enterprise-level, maintainable and robust mobile application" by Pawan Kumar and Salvatore Giordano
- Focuses on design patterns and best practices for Flutter app development, which is essential for creating a well-structured application.

CHAPTER 3

SYSTEM STUDY

3.1 EXISTING SYSTEM

The existing systems for expense management can vary widely, from manual methods like spreadsheets and physical receipts to more sophisticated software and applications. Here are some common approaches to expense management in the existing systems.

Manual Expense Tracking:

Many individuals and small businesses still rely on manual methods, such as recording expenses in a physical ledger or using spreadsheet software like Microsoft Excel or Google Sheets.

Physical receipts are collected and stored for later reconciliation with recorded expenses.

Bank Statements and Credit Card Statements:

Some individuals and businesses primarily rely on bank and credit card statements to track their expenses. These statements provide a record of all transactions, which can be categorized and reconciled manually.

Excel-based Templates:

Some individuals and businesses use pre-designed Excel templates for expense tracking. These templates offer a structured format for recording expenses and can be customized to some extent.

3.2 DISADVANTAGES OF EXISTING SYSTEM:

The existing systems for expense management, which often rely on manual or outdated processes, can have several disadvantages. These disadvantages can lead to inefficiencies, errors, and increased costs. Here are some common disadvantages of existing expense management systems:

Data Entry Errors:

Manual entry of expenses can lead to typographical errors, misplaced decimal points, and inaccuracies in financial records.

Time-Consuming:

Traditional methods like manual entry and paper-based processes are time-consuming, requiring significant effort to record, categorize, and reconcile expenses.

Loss of Receipts:

Physical receipts are easily lost or damaged, making it challenging to provide proof of expenses during audits or reimbursement requests.

Limited Accessibility:

Manual methods and paper-based systems limit accessibility to financial data, making it difficult to access and update records from multiple locations or devices.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 Software Description:

Front-End: Use Flutter to build a cross-platform mobile app, ensuring a consistent user experience on iOS and Android devices.

Backend: Utilize Firebase as the serverless backend for user authentication, real-time database, and cloud functions.

Authentication: Implement Google OAuth for secure and convenient user registration and login.

Notification Service (POST MVP) : Use a push notification service (e.g., Firebase Cloud Messaging or Apple Push Notification Service) to send notifications to users for reminders, updates, and messages.

4.2 List of Modules:

1. Authentication Module:

- User registration with Google OAuth.
- User login and logout functionality.

2. Dashboard Module:

- Display a list of existing rooms.
- Create a new room.
- Join existing rooms.
- Leave rooms.

3. Room Management Module:

- Room creation and deletion.
- Adding and removing members from a room.
- Displaying room details.

4. Expense Tracking Module:

- Expense recording with details (payee, participants, category, amount, date, etc.).
- Settlement of debts within a room.

5.Communication Module:

- Posting concerns or messages within a room.
- Real-time post functionality

6. User Management Module:

- User profiles with avatar and basic information.
- Inviting other users to join rooms.

4.3 Data Flow:

- Users log in or register using their Google accounts via the mobile app.
- Upon login, they access a dashboard displaying their existing rooms and can create new rooms.
- Inside each room, users can record expenses, track shared expenses, and communicate with room members through posts and chat.
- The system calculates and displays shared expenses for room members.
- Users can settle debts within rooms and view their financial history.

4.4 Libraries:

- **dynamic_color:** A library that makes it easier for your app to use dynamic color themes. You can change the color scheme in accordance with user preferences, app settings, or particular occasions. By giving your application a more unique look and feel, dynamic color customization can improve the overall user experience. Users can select color schemes that match their preferences, resulting in a setting that is more engaging and visually appealing environment.

- **firebase_core:** An essential library enables you to incorporate Firebase services into your Flutter application. It acts as Firebase's entry point, initializing the fundamental settings necessary for services like Firebase Authentication, Firestore, and Cloud Messaging. Through the use of firebase_core, you can make sure that your app connects to Firebase and is prepared to utilize the vast array of features it offers for user authentication, data storage, and real-time communication.

- **firebase_auth:** A library for handling user authentication in your app. It makes use of Firebase Authentication, a scalable and secure method of confirming user identities. To protect user accounts and ensure easy access to your ExpenseManager application, you can implement a number of authentication techniques, including email/password, Google sign-in, and more.

- **cloud_firestore:** A library that incorporate Firebase Cloud Firestore into your Flutter application. A NoSQL database called Firestore allows for effective data synchronization between devices and real-time data archiving. You can store and retrieve information about rooms, expenses, and chat messages using cloud_firestore, all while receiving real-time updates. For managing collaborative data in your app, it's a strong tool.

- **google_sign_in:** It makes it easier to integrate Google Sign-In into your Flutter application. Users can conveniently and securely log in or register using their Google accounts. Google Sign-In is a useful addition to your authentication procedures because it enjoys widespread user trust and use. It guarantees that users of your app have a smooth onboarding process.

- **fluttertoast:** It allows you to show toast notifications in your Flutter app. Toasts are discrete, transient messages that offer users feedback. You can display succinct notifications with fluttertoast, like success messages, error alerts, or straightforward confirmations. By giving users feedback on their actions, these notifications enhance user understanding and interaction with your app.

- **provider:** A state management library that plays a vital role in managing your app's state. It makes data sharing across the widget tree more effective and keeps data synchronized and current. This library makes the challenging process of managing data more straightforward and makes sure that your app reacts to user interactions and data changes efficiently.

- **email_validator:** A library that ensures that users provide accurate and correctly formatted email addresses during registration and when sending invitations within your app by determining whether email addresses comply with a valid email format.

- **connectivity:** A helpful for determining whether a user's device is connected to the internet. It enables your app to ascertain the user's online status or whether they are offline, which is crucial for real-time features like chat and data synchronization. With the help of this library, your app can respond to network availability in an intelligent way, improving user experience.

- **url_launcher:** Flutter app can launch external apps, send emails, and open web links thanks to url_launcher. Users' experiences are improved by the process's simplification of linking them to outside resources. You can enable users to access external apps, send emails, and visit web links from within your app by using url_launcher.

- **intl:** An essential for internationalization and localization support in your app. It empowers you to adapt your app to different languages, regions, and cultural contexts. With intl, you can localize your app's content, including dates, times, currency, and text, to create a global user experience that accommodates diverse audiences. It ensures that users from various regions can interact with your app in their preferred language and format.

4.5 User Interface:

1. Splash Screen:

A welcoming splash screen with the Expense Manager logo and a loading indicator, giving users a sense of a smooth start.

2. Login/Sign up Screen:

Google OAuth: A button to log in or sign-up using Google OAuth for secure and convenient authentication.

Registration Form: If a user is new, a registration form with fields for name, email, and password.

3. Dashboard:

Room List: A list of rooms the user is part of, displaying room names and avatars.
Create Room: A button or icon to create a new room.

Profile: An avatar or icon representing the user's profile, allowing them to access their personal information and settings.

4. Room Screen:

Room Name: Displaying the room name at the top.
Room Members: A section showing room members with avatars.
Room Description: A brief description of the room's purpose or context.
Expenses: A list of recorded expenses, including the payee, participants, amount, and category.
Expense Actions: Buttons or icons to add a new expense, settle debts, and view the expense history.
Posts Section: A chat-like section for posting concerns, messages, and general communication among room members.
Chat: A real-time chat section for discussing expenses, with avatars, timestamps, and message input.
Notifications: A notifications area for important updates or reminders related to the room.
Leave Room: A button to leave the room if the user wishes to exit.

5. Expense Entry Form:

Payee: A list is displayed to select the payee from room members.
Participants: A multi-select option or check boxes to choose participants.
Category: A dynamic input field is given for the purpose of payment.
Amount: An input field to enter the expense amount.
Save/Submit: Buttons to save or submit the expense.

4.6 Technical Implementation:

1. Set Up Flutter Project:

Create a new Flutter project or use an existing one as the foundation for your ExpenseManager app.

2. Firebase Integration:

Integrate Firebase into your Flutter project to handle authentication, data storage, and real-time synchronization. Use the firebase_core and firebase_auth libraries for this purpose.

3. User Authentication:

Implement user authentication using Firebase Authentication with Google OAuth. Allow users to log in with their Google accounts and create a new account if needed.

4. User Profile Management:

Develop screens for users to manage their profiles, including updating avatars, names, and emails. Use the Firebase Realtime Database or Firestore to store user profiles.

5. Room Management:

Create a data structure to manage rooms within the app. Use Firebase Realtime Database or Firestore to store room data, including room names, descriptions, and member lists.

6. Expense Recording:

Develop the UI and functionality for recording expenses within each room. Use Firebase Realtime Database or Firestore to store expense data, including details like payee, participants, amount, date, and category.

7. Expense Splitting:

Implement different expense splitting methods, such as equal splits, percentage splits, and custom shares. Calculate and display shared expenses for room members.

8. Chat and Communication:

Design the chat and communication features, allowing users to post messages and discuss expenses within each room. Use Firebase Realtime Database to enable real-time chat functionality.

9. Language and Localization:

Implement localization using the intl library to support multiple languages and regions, making the app accessible to a broader audience.

10. Settings and Preferences:

Develop settings screens that enable users to customize their app experience, such as language, theme, notification preferences, and data privacy settings.

11. Deployment:

Prepare your app for deployment on both iOS and Android platforms.

4.7 Testing:

A crucial step in the development process for an expense manager is testing. Some possible test types are listed below:

Unit tests:

Unit tests can be used to test each individual part of the system, including the model architecture, optimization algorithm, and data pre-processing. To make sure that each part of the system is operating properly, these tests can be quickly automated and run.

Integration Tests:

This testing phase ensures that various components, such as user interfaces and Firebase functions, integrate harmoniously, validating end-to-end functionality. Through simulated scenarios, developers assess the app's responsiveness to Firebase interactions, guaranteeing a robust and interconnected system.

Performance Tests:

Performance evaluations can be used to gauge how well a system performs under various loads, such as the volume of users or the size of the input images. These tests can be used to locate system bottlenecks and enhance performance.

User Acceptance Tests:

User acceptance tests can be used to evaluate the system from the standpoint of the end user. These tests can be used to check whether the program is simple to use, intuitive, and accurate.

Regression Tests:

Regression tests can be used to make sure that updates to the system don't result in the introduction of fresh bugs or regressions. To make sure that the system is reliable and stable over time, these tests can be used.

In general, testing is a crucial step in the creation of an expense manager project. To make sure that the system is operating properly, producing accurate results, and satisfying the needs of the end users, various tests can be used.

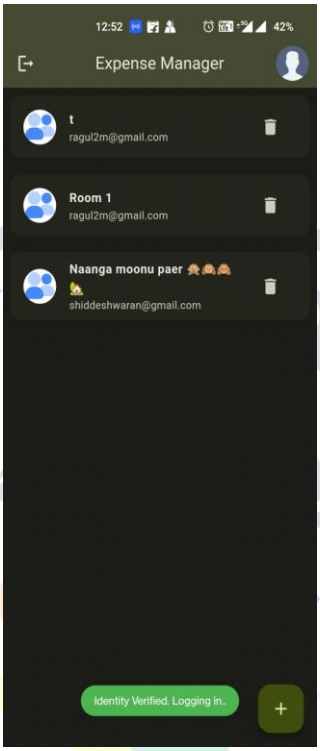
CHAPTER 5

WORKFLOW

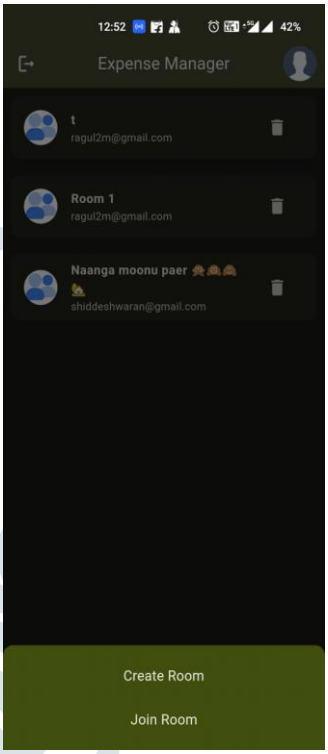
RESULT AND OUTPUT:

1) Sign in / Sign up page:

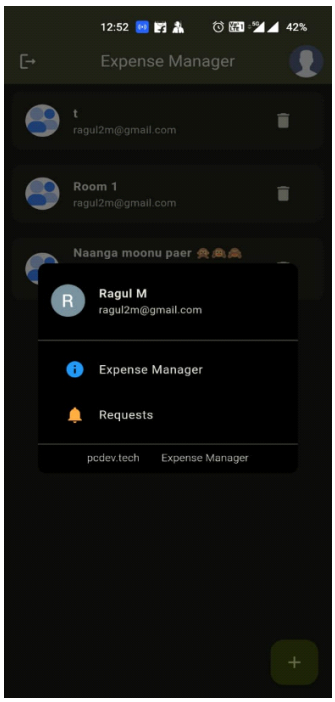
2) List of rooms



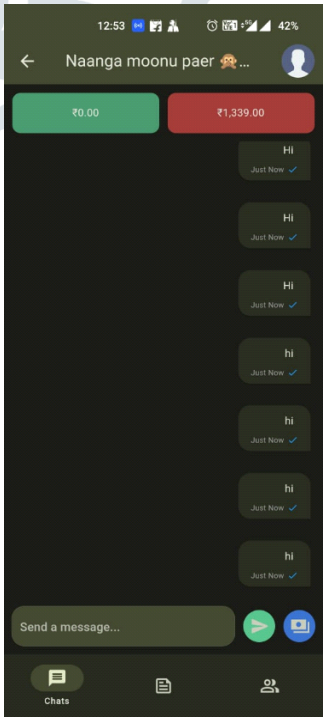
3) Create / Join room



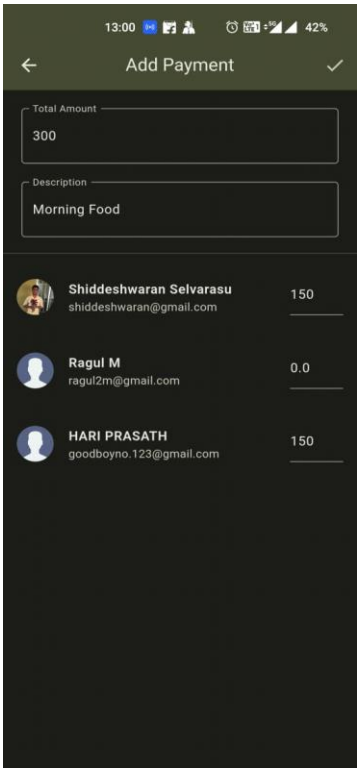
4) Signed in info page



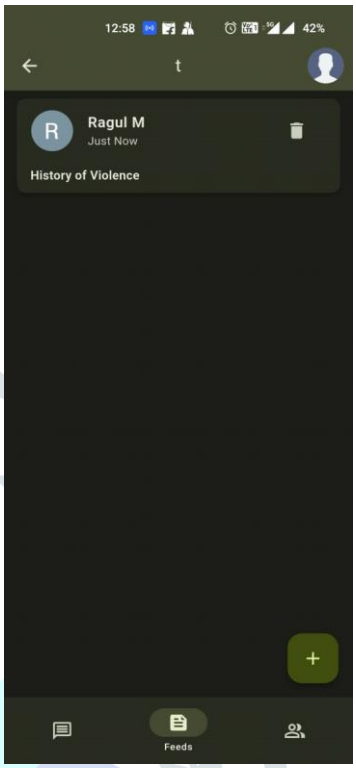
5) Chat Section



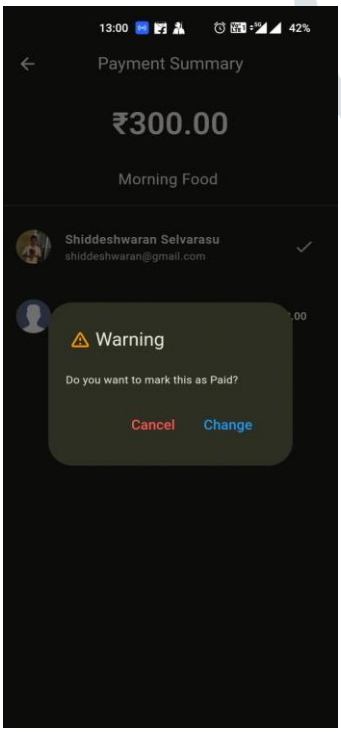
6)Feeds Section



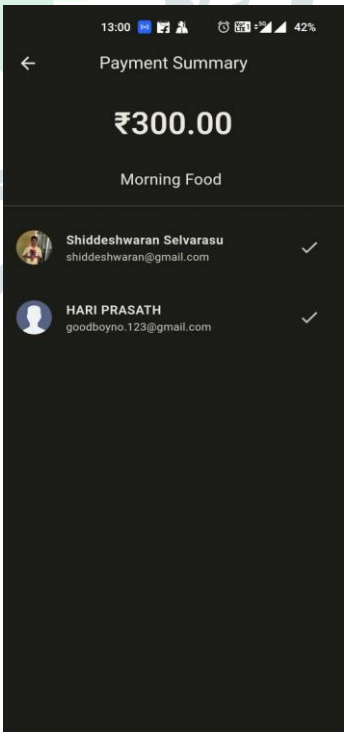
7) Creation of Expense



8)Making payee as paid



9)All paid status



CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION:

The Expense Manager project is a powerful and user-friendly mobile application designed to simplify and enhance group expense tracking and financial management for users. This collaborative expense manager offers a range of features to streamline the process of recording, splitting, and settling expenses among room members. With a focus on a clean and intuitive user interface, real-time communication, and internationalization, Expense Manager aims to provide an exceptional user experience.

The Expense Manager project is a valuable tool for a wide range of scenarios, from group travel expenses among friends to shared household costs and collaborative financial management among business colleagues. The ability to customize themes and the user-friendly approach to data management and communication make this application a versatile choice for users from various backgrounds.

In conclusion, the Expense Manager project represents a comprehensive solution for collaborative expense tracking, prioritizing user experience, security, and international appeal. Its continued development, supported by user feedback and the integration of emerging technologies, promises to deliver an even more robust and adaptable financial management tool for the future.

6.2 FUTURE SCOPE

Integration with Financial APIs:

Real-time updates on stock prices, currency exchange rates, and other financial indicators are provided through integration with financial APIs.

Expense Analytics Dashboard:

Developing a comprehensive analytics dashboard that graphically displays insights, categories, and spending trends.

Using interactive graphs and charts to make the experience more user-friendly.

Gamification for Budgeting:

Introduce gamification elements to motivate users to stick to their budgets and achieve financial goals.

Localization and Multi-Currency Support:

Enhance the app to support multiple languages and currencies for a global user base.

Expense Reminders and Alerts:

Developing a reminder system to alert users about upcoming bills, subscriptions, or budget milestones.

Financial Education Resources:

Inclusion of a section with educational resources on personal finance, budgeting tips, and investment basics.

Machine Learning for Spending Patterns:

Implementation of machine learning algorithms to analyze user spending patterns and provide personalized budgeting suggestions.

Predict future expenses based on historical data and user behavior.

Expense Receipt Scanning:

Inclusion of OCR (Optical Character Recognition) capabilities to scan and extract information from receipts, making expense logging more efficient.

APPENDIX
SOURCE CODE

chat.dart :

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

class Assignee {
  final String email;
  final double amount;
  bool status;

  Assignee.from({
    required this.email,
    required this.amount,
    required this.status,
  });

  Map<String, dynamic> toMap() {
    return {
      'assignee': email,
      'amount': amount.toString(),
      'status': status,
    };
  }

  static Assignee fromJson(var data) {
    return Assignee.from(
      email: data['assignee'],
      amount: double.parse(data['amount']),
      status: data['status'],
    );
  }
```

```
  @override
  String toString(){
    return toMap().toString();
  }

  String getPrice() {
    NumberFormat myFormat = NumberFormat.currency(
        locale: 'en_in', symbol: '\u{20B9}', decimalDigits: 2);
    // myFormat.minimumFractionDigits = 2;
    // myFormat.maximumSignificantDigits = 2;
    return myFormat.format(amount);
  }
}

class Message {
  final String id;
  final String sender;
  final String? description;
  final double totalAmount;
  final List<Assignee> assignees;
  final Timestamp createdDate;

  Message.from({
    required this.id,
    required this.sender,
    this.totalAmount = 0,
    required this.createdDate,
    this.assignees = const [],
    this.description,
  });

  addAssignees(Assignee assignee) {
    assignees.add(assignee);
  }

  String getPrice() {
    NumberFormat myFormat = NumberFormat.currency(
      locale: 'en_in',
      symbol: '\u{20B9}',
      decimalDigits: 2,
    );
    return myFormat.format(totalAmount);
  }

  Map<String, dynamic> toMap() {
    List<Map<String, dynamic>> list = [];
    for (var element in assignees) {
```

```
      list.add(element.toMap());
    }
    return {
      'sender': sender,
      'totalAmount': totalAmount.toString(),
      'description': description,
      'createdDate': createdDate,
      'assignees': list,
    };
  }

  static Message fromJson(var data,String id) {
    List<Assignee> assigneesList = [];
    if (data['assignees'] != null) {
      data['assignees'].forEach(
        (e) => assigneesList.add(Assignee.fromJson(e)),
      );
    }
    return Message.from(
      id: id,
      sender: data['sender'],
      totalAmount: double.parse(data['totalAmount'] ?? "0"),
      description: data['description'],
      createdDate: data['createdDate'],
      assignees: assigneesList,
    );
  }
}
```

feed.dart :

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

class Feed {
  String title;
  String? id;
  String imageURL;
  String description;
  Timestamp createdAt;
  String author;
  String link;
  String authorProfile;

  Feed.from({
    required this.title,
    required this.imageURL,
    required this.description,
    this.link = "",
```

```
    required this.createdAt,
    required this.author,
    required this.authorProfile,
  });

  Map<String, dynamic> toMap(){
    return {
      'title' : title,
      'imageURL': imageURL,
      'description': description,
      'createdAt': createdAt,
      'author' : author,
      'authorProfile': authorProfile,
    };
  }

  static Feed fromJson(var data) {
    return Feed.from(
      title: data['title'] ?? '',
      imageURL: data['imageURL'],
      description: data['description'],
      createdAt: data['createdAt'],
      author: data['author'],
      authorProfile: data['authorProfile'] ?? '',
    );
  }
}
```

room.dart :

```
import 'dart:math';

class Room {
  final String code;
  final String name;
  final String admin;
  final List<dynamic> members;

  Room.from({
    required this.code,
    required this.name,
    required this.admin,
    this.members = const [],
  });

  static Room createRoom({
    required String name,
```

```
    required String admin,
  }) {
    String charSet = DateTime.now().toString() + name + admin;
    charSet = charSet.replaceAll(RegExp(r'[^\w]+'), '');
    Random ram = Random();
    List<String> list = [];
    list.add(admin);
    var code = String.fromCharCodes(
      Iterable.generate(
        8,
        (_) => charSet.codeUnitAt(ram.nextInt(charSet.length)),
      ),
    );
    return Room.from(code: code, name: name, admin: admin, members: list);
  }

  @override
  bool operator ==(Object other) =>
      identical(this, other) || other is Room && code == other.code;

  Map<String, dynamic> toJson() {
    return {
      'name': name,
      'admin': admin,
      'people': members,
      'code': code,
    };
  }

  static Room fromJson(Map data) {
    return Room.from(
      name: data['name'],
      admin: data['admin'],
      members: data['people'],
      code: data['code'],
    );
  }
}
```

about_page.dart :

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

class AboutPage extends StatelessWidget {
  const AboutPage({Key? key}) : super(key: key);

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

```
    var brightness = MediaQuery.of(context).platformBrightness;
    return Scaffold(
      appBar: AppBar(
        elevation: 1,
        centerTitle: true,
        title: Text(
          "Expanse Manager",
          style: TextStyle(
            color: brightness == Brightness.light ? Colors.black : Colors.white,
          ),
        ),
        backgroundColor:
        brightness == Brightness.dark ? Colors.black26 : Colors.white,
        leading: IconButton(
          color: brightness == Brightness.light ? Colors.black : Colors.white,
          onPressed: () {
            Navigator.pop(context);
          },
          icon: const Icon(Icons.close),
        ),
      ),
      body: Center(
        child: Image.asset('assets/dash.png'),
      ),
    );
  }
}
```

add_payment.dart :

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:expense_manager/models/chat.dart';
import 'package:expense_manager/models/constants.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';

import '../models/room.dart';

class AddPayment extends StatefulWidget {
  const AddPayment({required this.room, super.key});

  final Room room;

  @override
  State<AddPayment> createState() => _AddPaymentState();
}
```

```
class _AddPaymentState extends State<AddPayment> {
  final user = FirebaseAuth.instance.currentUser;
  final _amountController = TextEditingController();
  final _descriptionController = TextEditingController();
  String? amountError;
  final List<AssigneeTextField> _assigneeControllers = [];
  double defaultVal = 0.0;
  NumberFormat myFormat = NumberFormat.currency(
    locale: 'en_in',
    symbol: '\u{20B9}',
    decimalDigits: 2,
  );

  Widget assigneeCard(UserData userData, int index) {
    if (_assigneeControllers.length <= index) {
      _assigneeControllers.add(
        AssigneeTextField(
          email: userData.email,
          controller: TextEditingController(),
        ),
      );
    }
    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
      child: StreamBuilder(
        stream: FirebaseFirestore.instance
            .collection('users')
            .doc(userData.email)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            UserData userData = UserData.fromMap(snapshot.data!.data()!);
            return ListTile(
              leading: CircleAvatar(
                foregroundImage: NetworkImage(userData.profileImageUrl),
              ),
              title: Text(
                userData.name,
                style: const TextStyle(fontWeight: FontWeight.w600),
                overflow: TextOverflow.ellipsis,
              ),
              subtitle: Text(userData.email),
              contentPadding: EdgeInsets.zero,
              trailing: SizedBox(
                height: 60,
                width: MediaQuery.of(context).size.width * 0.15,
                child: TextField(
                  controller: _assigneeControllers[index].controller,
```

```
              decoration: InputDecoration(
                errorText: _assigneeControllers[index].error,
              ),
              onChanged: (txt) {
                if (txt.isEmpty) {
                  _assigneeControllers[index].controller.text = '0.0';
                }
                setState(() {
                  defaultVal = double.parse('0.0');
                });
              },
            ),
          ),
        );
      }
      return const SizedBox();
    },
  ),
 );
}

sendMessage() {
 List<Assignee> assignees = [];

 for (AssigneeTextField assTxt in _assigneeControllers) {
  if(assTxt.getVal() == 0.0){
   continue;
  }
  assignees.add(
   Assignee.from(
    email: assTxt.email,
    amount: assTxt.getVal(),
    status: assTxt.email == user!.email!,
   ),
  );
 }

 Message message = Message.from(
  id: 'gfhfhg',
  sender: user!.email!,
  createdDate: Timestamp.now(),
  assignees: assignees,
  description: _descriptionController.value.text,
  totalAmount: double.parse(_amountController.value.text),
 );

 FirebaseFirestore.instance.collection('rooms/${widget.room.code}/chats').add(message.toMap());
}
```

```
getDiff() {
  double total = defaultVal;
  try {
    total = double.parse(_amountController.value.text);
  } catch (e) {
    print(e);
  }
  double sum = 0.0;

  for (AssigneeTextField assTxt in _assigneeControllers) {
    sum += assTxt.getVal();
  }

  return total - sum;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Theme.of(context).colorScheme.background,
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.secondaryContainer,
      centerTitle: true,
      title: const Text('Add Payment'),
      actions: [
        IconButton(
          onPressed: () {
            if (_amountController.value.text.isNotEmpty &&
                _assigneeControllers.isNotEmpty) {
              Navigator.pop(context);
              sendMessage();
            } else if (_amountController.value.text.isEmpty) {
              setState(() {
                amountError = 'Amount can\'t be empty';
              });
            }
          },
          icon: const Icon(Icons.done),
        ),
      ],
    ),
    body: SingleChildScrollView(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisSize: MainAxisSize.max,
        children: [
```

```
SizedBox(
  child: Center(
    child: Padding(
      padding: const EdgeInsets.only(
          top: 20, left: 20, right: 20, bottom: 10),
      child: TextField(
        controller: _amountController,
        keyboardType: TextInputType.number,
        decoration: InputDecoration(
          border: const OutlineInputBorder(
            borderSide: BorderSide(),
          ),
          errorText: amountError,
          hintText: 'Enter Total Amount',
          labelText: 'Total Amount',
        ),
        onChanged: (text) {
          setState(() {
            if (text.isNotEmpty) {
              amountError = null;
            }
            defaultVal = 0.0;
          });
          double val = 0.0;
          if (text.isNotEmpty) {
            val = double.parse(text) / widget.room.members.length;
          }
          for (AssigneeTextField assTxt in _assigneeControllers) {
            assTxt.changeVal(val);
          }
        },
      ),
    ),
  ),
),
SizedBox(
  child: Center(
    child: Padding(
      padding:
          const EdgeInsets.symmetric(horizontal: 20, vertical: 10),
      child: TextField(
        controller: _descriptionController,
        keyboardType: TextInputType.text,
        decoration: const InputDecoration(
          border: OutlineInputBorder(
            borderSide: BorderSide(),
          ),
          hintText: 'Enter description',
```

```
                labelText: 'Description',
              ),
            ),
          ),
        ),
      ),
      const Divider(),
      ListView.builder(
        physics: const NeverScrollableScrollPhysics(),
        shrinkWrap: true,
        itemCount: widget.room.members.length,
        itemBuilder: (context, index) {
          return StreamBuilder(
            stream: FirebaseFirestore.instance
                .collection('users')
                .doc(widget.room.members[index])
                .snapshots(),
            builder: (context, snapshot) {
              if (snapshot.hasData) {
                UserData userDate =
                    UserData.fromMap(snapshot.data!.data()!);
                return assigneeCard(userDate, index);
              }
              return const SizedBox();
            },
          );
        },
      ),
      if (getDiff() != 0.0)
        Text(
          'Difference Amount: ${myFormat.format(getDiff().abs())}',
          style: TextStyle(
            fontWeight: FontWeight.bold,
            color: getDiff() < 0 ? Colors.green : Colors.red,
          ),
        ),
    ],
  ),
 ),
 );
 }
}
```

add_people.dart :

```
import 'dart:math';

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

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

import '../models/helper.dart';

class AddPeople extends StatefulWidget {
  const AddPeople({Key? key, required this.code}) : super(key: key);

  final String code;

  @override
  State<AddPeople> createState() => _AddPeopleState();
}

class _AddPeopleState extends State<AddPeople> {
  List<AddPeopleList> conList = [
    AddPeopleList(controller: TextEditingController(), key: "getUniqueKey()")
  ];

  void _showToast(String text) {
    Fluttertoast.showToast(
      msg: text,
      gravity: ToastGravity.BOTTOM,
      toastLength: Toast.LENGTH_LONG,
      timeInSecForIosWeb: 250,
    );
  }

  getUniqueKey() {
    String charSet =
        '${DateTime.now()}AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz';
    charSet = charSet.replaceAll(RegExp(r'[^\w]+'), '');
    Random ram = Random();
    return String.fromCharCodes(Iterable.generate(
        25, (_) => charSet.codeUnitAt(ram.nextInt(charSet.length))));
  }

  addEmail() {
    setState(() {
      conList.add(AddPeopleList(
          controller: TextEditingController(), key: getUniqueKey()));
    });
  }

  checkFields() {
    var count = 0;
    for (var element in conList) {
```

```
    var text = element.controller.value.text;
    if (EmailValidator.validate(text)) {
      setState(() {
        element.errorText = 'Enter a Valid Email!';
      });
    } else {
      count++;
    }
  }
  if (count == 0) {
    return true;
  } else {
    return false;
  }
}

updateUser(List<dynamic> addList) {
  for (AddPeopleList e in addList) {
    var snapshot =
        FirebaseFirestore.instance.doc('/users/${e.controller.value.text}');
    snapshot.get().then((value) {
      if (value.exists) {
        snapshot.update({
          (e.isOptional ? 'requests' : 'rooms'): FieldValue.arrayUnion(
            [FirebaseFirestore.instance.doc('/rooms/${widget.code}')],
          ),
        });
      }
    });
  }
}

addPeople() {
  if (checkFields()) {
    List<dynamic> addList = [];
    List<dynamic> addEmailList = [];
    for (var element in conList) {
      if (EmailValidator.validate(element.controller.value.text)) {
        addEmailList.add(element.controller.value.text);
        addList.add(element);
      }
    }
    FirebaseFirestore.instance.doc('/rooms/${widget.code}').update({
      'people': FieldValue.arrayUnion(addEmailList),
    }).then((value) {
      updateUser(addList);
      _showToast("Members Added! Refresh the page to See the changes.");
    }).onError((error, stackTrace) {
```

```
      _showToast(error.toString());
    });
    return true;
  } else {
    return false;
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      title: const Text(
        "Add People",
      ),
      backgroundColor: Theme.of(context).colorScheme.secondaryContainer,
      leading: IconButton(
        onPressed: () {
          Navigator.pop(context);
        },
        icon: const Icon(Icons.close),
      ),
      actions: [
        IconButton(
          onPressed: () {
            if (addPeople()) {
              Navigator.pop(context);
              Navigator.pop(context);
            }
          },
          icon: const Icon(Icons.check),
        ),
      ],
    ),
    body: conList.isEmpty
        ? const Center(
            child: Text("Click the Add button to start Adding people."),
          )
        : ListView.builder(
            itemCount: conList.length,
            itemBuilder: (context, index) {
              return Dismissible(
                key: Key(conList[index].key),
                onDismissed: (_) {
                  setState(() {
                    conList.remove(conList[index]);
                  });
```

```
},
background: Container(color: Colors.red),
child: Padding(
 padding: const EdgeInsets.all(10),
 child: ListTile(
  leading: IconButton(
   onPressed: () {
    setState(() {
     if (conList[index].isOptional) {
      conList[index].isOptional = false;
     } else {
      conList[index].isOptional = true;
     }
    });
   },
   icon: Icon(
    Icons.email_rounded,
    color: conList[index].isOptional
      ? Colors.blue
      : Colors.blueGrey,
    size: 30,
   ),
  ),
  title: TextField(
   controller: conList[index].controller,
   decoration: InputDecoration(
    hintText: 'Enter Member Email',
    labelText: 'Member Email',
    errorText: conList[index].errorText,
    border: const OutlineInputBorder(
     borderSide: BorderSide(),
    ),
   ),
   keyboardType: TextInputType.emailAddress,
   onChanged: (text) {
    if (EmailValidator.validate(text)) {
     setState(() {
      conList[index].errorText = null;
     });
    } else {
     setState(() {
      conList[index].errorText = 'Enter a Valid Email!';
     });
    }
   },
  ),
 ),
),
```

```
              );
            }),
      floatingActionButton: FloatingActionButton(
        onPressed: addEmail,
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

add_room.dart:

```
import 'dart:math';

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

import '../models/room.dart';

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

  @override
  State<AddRoom> createState() => _AddRoomState();
}

class _AddRoomState extends State<AddRoom> {
  final user = FirebaseAuth.instance.currentUser;
  final TextEditingController _nameCon = TextEditingController();
  String? nameErrText = null;

  void _showToast(String text, Color color) {
    Fluttertoast.showToast(
      msg: text,
      backgroundColor: color,
      gravity: ToastGravity.BOTTOM,
      toastLength: Toast.LENGTH_LONG,
      timeInSecForIosWeb: 250,
    );
  }

  int getRandomImage() {
    Random ram = Random();
    return ram.nextInt(9);
  }
```

```
createClassRoom() {
  var room = Room.createRoom(
    name: _nameCon.value.text,
    admin: user!.email!,
  );
  if (_nameCon.value.text != '') {
    var snapshot =
        FirebaseFirestore.instance.collection('/rooms').doc(room.code);
    snapshot.get().then((value) {
      if (value.exists) {
        _showToast("something Went Wrong Try Again!", Colors.redAccent);
      } else {
        snapshot.set(room.toJson());
        _showToast("Room Created", Colors.greenAccent);
      }
    });
    FirebaseFirestore.instance.collection('/users').doc(user!.email).update({
      'rooms': FieldValue.arrayUnion([snapshot]),
    });
    Navigator.pop(context);
  } else {
    setState(() {
      nameErrText = 'Name can\'t be Empty';
    });
    _showToast("Name Field Cannot be Empty", Colors.redAccent);
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      centerTitle: true,
      title: const Text(
        "Add Room",
      ),
      backgroundColor: Theme.of(context).colorScheme.primaryContainer,
      leading: IconButton(
        onPressed: () {
          Navigator.pop(context);
        },
        icon: const Icon(Icons.close),
      ),
      actions: [
        IconButton(
          onPressed: () {
            createClassRoom();
          },
```

```
          icon: const Icon(Icons.check),
        ),
      ],
    ),
    body: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 15),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 10),
            child: TextField(
              controller: _nameCon,
              decoration: InputDecoration(
                border: const OutlineInputBorder(
                  borderSide: BorderSide(),
                ),
                errorText: nameErrText,
                hintText: 'Enter Room Name',
                labelText: 'Room Name',
              ),
              onChanged: (text) {
                if (text == '') {
                  setState(() {
                    nameErrText = 'Name can\'t be Empty';
                  });
                } else {
                  setState(() {
                    nameErrText = null;
                  });
                }
              },
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

feed_view.dart :

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

import '../models/feed.dart';
import '../utils/time_handler.dart';
```

```
class FeedView extends StatelessWidget {
  const FeedView({Key? key, required this.feed}) : super(key: key);

  final Feed feed;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        elevation: 0,
        centerTitle: true,
        title: const Text("Feed"),
        backgroundColor: Theme.of(context).colorScheme.secondaryContainer,
        leading: IconButton(
          onPressed: () {
            Navigator.pop(context);
          },
          icon: const Icon(Icons.close),
        ),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 15),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                  ListTile(
                    leading: CircleAvatar(
                      foregroundImage: NetworkImage(feed.authorProfile),
                    ),
                    title: Text(
                      feed.author,
                      style: const TextStyle(fontWeight: FontWeight.w500),
                    ),
                    subtitle: Text(
                        DateTimeHandler.getDateTimeDiff(feed.createdAt.toDate())),
                  ),
                  const Divider(
                    thickness: 1.2,
                  ),
                  Padding(
                    padding: const EdgeInsets.all(10),
                    child: Text(
```

```
                feed.title,
                style: const TextStyle(
                  fontWeight: FontWeight.w600,
                  fontSize: 20,
                ),
              ),
            ),
          ],
        ),
        const SizedBox(
          height: 15,
        ),
        feed.imageURL != ''
            ? Image.network(feed.imageURL)
            : const SizedBox(),
        Padding(
          padding:
              const EdgeInsets.symmetric(vertical: 20, horizontal: 10),
          child: Text(feed.description, textAlign: TextAlign.justify),
        ),
        feed.link != ''
            ? TextButton(
                onPressed: () {},
                child: Text(
                  feed.link,
                  style: const TextStyle(color: Colors.blue),
                ))
            : const SizedBox(),
      ],
    ),
  ),
),
);
}
}
```

home_page.dart:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:expense_manager/models/constants.dart';
import 'package:expense_manager/pages/add_room.dart';
import 'package:expense_manager/pages/request_page.dart';
import 'package:expense_manager/utils/login_manager.dart';
import 'package:expense_manager/widgets/rooms_list.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
```

```
import '../models/room.dart';
import '../widgets/account_popup.dart';
import '../widgets/popup_dialog.dart';

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final user = FirebaseAuth.instance.currentUser;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).colorScheme.background,
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.secondaryContainer,
        centerTitle: true,
        title: const Text('Expense Manager'),
        leading: IconButton(
          onPressed: () {
            final provider =
                Provider.of<LoginProvider>(context, listen: false);
            provider.logout();
          },
          icon: const Icon(Icons.logout_rounded),
        ),
        actions: [
          StreamBuilder(
              stream: FirebaseFirestore.instance
                  .collection('users')
                  .doc(user!.email)
                  .snapshots(),
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  var data = snapshot.data!.data();
                  var photo = data!['profileImageUrl'] ?? defaultProfileImage;
                  return Padding(
                    padding: const EdgeInsets.symmetric(
                        horizontal: 10, vertical: 5),
                    child: InkWell(
                      onTap: () {
                        showDialog(
                          context: context,
                          builder: (context) => AccountPopUp(
```

```
                action: [
                  Padding(
                    padding: const EdgeInsets.symmetric(
                        vertical: 0, horizontal: 15),
                    child: ListTile(
                      onTap: () {
                        Navigator.pop(context);
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) =>
                                const RequestPage(),
                          ),
                        );
                      },
                      leading: const Icon(
                        Icons.notifications,
                        color: Colors.orangeAccent,
                      ),
                      title: const Text("Requests"),
                    ),
                  ),
                ],
              ),
            );
          },
          child: CircleAvatar(
            backgroundColor: Theme.of(context)
                .colorScheme
                .primary
                .withAlpha(50),
            foregroundImage: NetworkImage(photo),
          ),
        ),
      );
    }
    return const Padding(
      padding: EdgeInsets.symmetric(horizontal: 10),
      child: CircleAvatar(
        child: CircularProgressIndicator(),
      ),
    );
  }),
 ],
),
body: const RoomsList(),
floatingActionButton: FloatingActionButton(
 onPressed: () {
```

```
        _showBottomSheet();
      },
      child: const Icon(Icons.add),
    ));
}

_showBottomSheet() {
  return showModalBottomSheet(
    context: context,
    isScrollControlled: true,
    backgroundColor: Colors.transparent,
    builder: (BuildContext context) {
      return SingleChildScrollView(
        child: AnimatedPadding(
          duration: const Duration(milliseconds: 150),
          curve: Curves.easeOut,
          padding: EdgeInsets.only(
              bottom: MediaQuery.of(context).viewInsets.bottom),
          child: Container(
            decoration: BoxDecoration(
              borderRadius: const BorderRadius.only(
                topLeft: Radius.circular(12),
                topRight: Radius.circular(12),
              ),
              color: Theme.of(context).colorScheme.primaryContainer,
            ),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.start,
              mainAxisSize: MainAxisSize.min,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                const SizedBox(
                  height: 10,
                ),
                Padding(
                  padding:
                      const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
                  child: InkWell(
                    splashColor: Colors.grey,
                    onTap: () {
                      Navigator.pop(context);
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => const AddRoom(),
                        ),
                      );
                    },
```

```
            child: SizedBox(
              width: MediaQuery.of(context).size.width - 15,
              height: 40,
              child: const Center(
                child: Text(
                  "Create Room",
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.w400,
                  ),
                ),
              ),
            ),
          ),
        ),
        Padding(
          padding:
              const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
          child: InkWell(
            onTap: () async {
              Navigator.pop(context);
              await showDialog(
                context: context,
                builder: (context) => const PopUpDialog(),
              );
            },
            child: SizedBox(
              width: MediaQuery.of(context).size.width - 15,
              height: 40,
              child: const Center(
                child: Text(
                  "Join Room",
                  style: TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.w400,
                  ),
                ),
              ),
            ),
          ),
        ),
        const SizedBox(
          height: 10,
        ),
      ],
    ),
  ),
),
```

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

login_page.dart :

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:expense_manager/pages/signup_page.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:email_validator/email_validator.dart';

import '../utils/login_manager.dart';

class LoginPage extends StatefulWidget {
  const LoginPage({required this.random, Key? key}) : super(key: key);

  final int random;

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();
  String? emailError;
  String? passError;

  @override
  Widget build(BuildContext context) {
    final provider = Provider.of<LoginProvider>(context, listen: false);
    return Scaffold(
      resizeToAvoidBottomInset: true,
      appBar: AppBar(
        centerTitle: true,
        title: const Text('Expense Manager'),
        backgroundColor: Theme.of(context).colorScheme.secondaryContainer,
      ),
      backgroundColor: Theme.of(context).colorScheme.background,
      body: Column(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
```

```
Expanded(
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 60, vertical: 20),
    child: Image.asset('assets/images/${widget.random}.png'),
  ),
),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 60, vertical: 10),
  child: TextField(
    controller: email,
    decoration: InputDecoration(
      hintText: 'Enter Email',
      errorText: emailError,
    ),
    keyboardType: TextInputType.emailAddress,
    autofillHints: const [
      AutofillHints.username,
      AutofillHints.newUsername
    ],
    onChanged: (text) {
      setState(() {
        if (EmailValidator.validate(text)) {
          emailError = null;
        } else {
          emailError = 'Enter a Valid Email';
        }
      });
    },
  ),
),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 60, vertical: 10),
  child: TextField(
    controller: password,
    decoration: InputDecoration(
      hintText: 'Enter password',
      errorText: passError,
    ),
    obscureText: true,
    enableSuggestions: false,
    autocorrect: false,
    autofillHints: const [
      AutofillHints.password,
      AutofillHints.newPassword
    ],
    onChanged: (text) {
      setState(() {
        if (text.length >= 8) {
```

```
          passError = null;
        } else {
          passError = 'Enter At least 8 characters';
        }
      });
    },
    keyboardType: TextInputType.visiblePassword,
  ),
),
Center(
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 0),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      mainAxisSize: MainAxisSize.max,
      children: [
        TextButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const SignUpPage(),
              ),
            );
          },
          child: const Text('Create account'),
        ),
        TextButton(
          onPressed: () {},
          child: const Text('Forgot password?'),
        ),
      ],
    ),
  ),
),
SizedBox(
  width: MediaQuery.of(context).size.width,
  height: 90,
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),
    child: ElevatedButton(
      onPressed: () {
        if ((emailError == null || email.value.text == '') &&
            (passError == null || password.value.text == '')) {
          provider.login(email.value.text, password.value.text);
        }
      },
      style: ElevatedButton.styleFrom(
```

```
              backgroundColor:
                  Theme.of(context).colorScheme.primaryContainer,
            ),
            child: const Text('Login'),
          ),
        ),
      ),
      SizedBox(
        width: MediaQuery.of(context).size.width,
        height: 90,
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 50, vertical: 20),
          child: OutlinedButton.icon(
              label: const Text(
                'Sign in with Google',
                style: TextStyle(color: Colors.black,fontSize: 18,),
              ),
              style: OutlinedButton.styleFrom(
                backgroundColor: Colors.white,
                elevation: 1,
                shape: const StadiumBorder(),
                side: const BorderSide(
                    width: 2, color: Colors.black12),
              ),
              icon: Padding(
                padding: const EdgeInsets.only(top: 8,bottom: 8,),
                child: Image.asset(
                  'assets/google-icon.png',
                  width: 35,
                ),
              ),
              onPressed: () {
                provider.googleLogIn();
              }),
        ),
      ),
    ],
  ),
 );
 }
}
```

message_detail.dart:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:expense_manager/models/chat.dart';
import 'package:expense_manager/models/constants.dart';
```

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

import '../models/room.dart';

class MessageDetail extends StatefulWidget {
  const MessageDetail({required this.message, required this.room, super.key});

  final Message message;
  final Room room;

  @override
  State<MessageDetail> createState() => _MessageDetailState();
}

class _MessageDetailState extends State<MessageDetail> {
  final user = FirebaseAuth.instance.currentUser;

  showAlert(bool status, int index) {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: const Row(
            children: [
              Icon(
                Icons.warning_amber_rounded,
                color: Colors.orange,
                size: 25,
              ),
              Padding(
                padding: EdgeInsets.all(5),
                child: Text("Warning"),
              ),
            ],
          ),
          contentPadding: const EdgeInsets.all(20),
          content: Text(
              "Do you want to mark this as ${status ? 'Unpaid' : 'Paid'}?"),
          actionsAlignment: MainAxisAlignment.end,
          elevation: 5,
          actions: [
            TextButton(
              onPressed: () => Navigator.pop(context),
              child: const Text(
                "Cancel",
                style: TextStyle(
                  color: Colors.redAccent,
```

```
            fontSize: 18,
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(right: 10),
        child: TextButton(
          onPressed: () {
            Navigator.pop(context);
            setState(() {
              widget.message.assignees[index].status = !status;
            });
            FirebaseFirestore.instance
                .collection('rooms/${widget.room.code}/chats')
                .doc(widget.message.id)
                .update(widget.message.toMap());
          },
          child: const Text(
            "Change",
            style: TextStyle(
              color: Colors.blue,
              fontSize: 18,
            ),
          ),
        ),
      ),
    ],
  );
 },
);
}

Widget assigneeCard(Assignee assignee, int index) {
 return Padding(
  padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
  child: StreamBuilder(
   stream: FirebaseFirestore.instance
     .collection('users')
     .doc(assignee.email)
     .snapshots(),
   builder: (context, snapshot) {
    if (snapshot.hasData) {
     UserData userData = UserData.fromMap(snapshot.data!.data()!);
     return ListTile(
      leading: CircleAvatar(
       foregroundImage: NetworkImage(userData.profileImageUrl),
      ),
      title: Text(
```

```
            userData.name,
            style: const TextStyle(fontWeight: FontWeight.w600),
            overflow: TextOverflow.ellipsis,
          ),
          subtitle: Text(userData.email),
          contentPadding: EdgeInsets.zero,
          trailing: assignee.status
              ? IconButton(
                  onPressed: () {
                    if(widget.message.sender == user!.email) {
                      showAlert(assignee.status, index);
                    }
                  },
                  icon: const Icon(Icons.done),
                )
              : MaterialButton(
                  onPressed: () {
                    if(widget.message.sender == user!.email) {
                      showAlert(assignee.status, index);
                    }
                  },
                  child: Text(assignee.getPrice()),
                ),
        );
      }

      return const SizedBox();
    },
  ),
 );
}

@override
Widget build(BuildContext context) {
 return Scaffold(
  backgroundColor: Theme.of(context).colorScheme.background,
  appBar: AppBar(
   centerTitle: true,
   title: const Text('Payment Summary'),
  ),
  body: SingleChildScrollView(
   child: Column(
    crossAxisAlignment: CrossAxisAlignment.center,
    mainAxisSize: MainAxisSize.max,
    mainAxisAlignment: MainAxisAlignment.start,
    children: [
     SizedBox(
      height: MediaQuery.of(context).size.height * 0.1,
```

```
            child: Center(
              child: Text(
                widget.message.getPrice(),
                style: const TextStyle(
                  fontSize: 40,
                  fontWeight: FontWeight.w600,
                ),
              ),
            ),
          ),
          SizedBox(
            child: Center(
              child: Padding(
                padding: const EdgeInsets.all(10),
                child: Text(
                  widget.message.description ?? '',
                  style: const TextStyle(
                    fontSize: 20,
                  ),
                ),
              ),
            ),
          ),
          const Divider(),
          ListView.builder(
            physics: const NeverScrollableScrollPhysics(),
            shrinkWrap: true,
            itemCount: widget.message.assignees.length,
            itemBuilder: (context, index) {
              var assignee = widget.message.assignees[index];
              return assigneeCard(assignee, index);
            },
          ),
        ],
      ),
    ),
  );
 }
}
```

REFERENCES

Flutter Documentation:

https://docs.flutter.dev/

Official Flutter Documentation - The official Flutter documentation provides in-depth information on Flutter's features, widgets, and best practices.

Firebase Documentation:

https://firebase.google.com/docs

Firebase Documentation - The Firebase documentation offers a wealth of information on integrating Firebase services into your Flutter project.

Google OAuth Integration:

https://firebase.google.com/docs/auth

Firebase Authentication with Google - Learn how to integrate Google OAuth for user authentication in Firebase.

Dart Programming:

https://dart.dev/guides

Dart Documentation - Dart is the programming language used with Flutter. The Dart documentation covers language features and best practices.

State Management with Provider:

https://docs.flutter.dev/data-and-backend/state-mgmt/simple

Provider Package - The official documentation for the provider package explains state management in Flutter.

Internationalization and Localization:

https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization

Internationalization (i18n) - Learn about internationalization and localization in Flutter.

Firebase Realtime Database and Firestore:

https://firebase.google.com/docs/database/rtdb-vs-firestore

Firebase Realtime Database - For information on using Firebase Realtime Database.

Cloud Firestore - For details on using Firebase Firestore for data storage.

App Deployment and Publishing:

https://developer.android.com/studio/publish

Publishing to the Play Store - For publishing your Android app on Google Play.

Research on best practices for securing user data and protecting privacy within mobile apps.