**Assignment 3**

**Money Sharing Application**

Github repository link:<https://github.com/SoftDevMobDev-2024-Classrooms/assignment03-TranVuDuc04>

Firebase link: <https://console.firebase.google.com/u/0/project/moneyshare-e2a0f/database/moneyshare-e2a0f-default-rtdb/data>

**Introduction**

This is a simple app called Money Sharing Application, it is a user-friendly mobile application designed to streamline financial interactions among friends and event participants. The app allows users to create events, track participants, and manage shared expense. With features like payment tracking, participant management, MoneyShare ensures transparency and ease in group spending. It receives feedbacks and allows both portrait and landscape orientation for users.

**Development plan**

1. **Requirements Analysis:** CRUD functionalities (participant management, event organization, and financial tracking for the MoneyShare app), log demonstration, Ui Testing, use of RecyclerView, database Firebase with CRUD functionality, app’s complexity, components and advance components.

**How to use**

This app provides an experience for user who wants calculating and share the spendings for his family or a group of friends. The user can create multiple separate events with multiple participants and multiple spendings. The application will automatically calculates the total amount of money each participant owes or is owed, each participants will be displayed with different colours responding to their money status. The app manages data in Firebase. User can even delete any event. After deleting, the app will ask for a feedback and this feedback can be seen in a separate section.

**User stories and sketches**

**User Story 1**: **As a participant**, I want to add my payment details easily so that I can track my expenses for the event without confusion. I expect to see a clear and simple input form and receive immediate confirmation of my submission.

**Use Case:** The participant is on his payment page, then clicks the add button. A simple form is displayed.

**A screenshot of a mobile application

Description automatically generated**

Figure 1. Sketch 1

**User Story 2: As an event organizer,** I want to view all participants and their payment statuses so that I can ensure everyone has contributed fairly. I need a summary of total amounts owed and paid by each participant to facilitate financial management.

**Use Case:** The organizer creates an event, adds total participants and input all their spending.

A screenshot of a phone

Description automatically generated

Figure 2. Sketch 2

**2. Design**

* UI Layout: Create displays for events, participants,payment summaries, and feedback.
* Buttons: features for adding participants, payments, and navigating.
* App Logic: Implement participants and financial calculations, total amounts owed, and dynamic updates during data changes.

MainActivity: Allow user to navigate to Feedback section or Create new events. By pressing create button, a form is shown for user to input (UploadActivity). When an event is displayed, the user can access more detail of that event.

|  |  |
| --- | --- |
| Figure 3. MainActivity | A screenshot of a phone  Description automatically generated  Figure 4. UploadActivity |

ParticipantActivity: Shows the total number of participants and total amount of money spent in an event. User can add more participants. The financial status of each participants affects the display of them with corresponding colours.   
Two different scenerios:

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated  Figure 5. Scenario 1 | Figure 6. Scenario 2 |

PaymentActivity: Shows the total money that participants spent along with each payment they made with description. When add button is pressed, AddPaymentActivity displays as a simple form for easy input.

|  |  |
| --- | --- |
| Figure 7. PaymentActivity | A screen shot of a phone  Description automatically generated  Figure 8. AddPaymentActivity |

FeedbackActivity: After deleting an event, a dialog (FeedbackFragment) will pop up to get user’s feedback. User can later access the latest feedback via a button in MainActivity.

|  |  |
| --- | --- |
| Figure 9. FeedbackFragment | A screen shot of a phone  Description automatically generated  Figure 10. FeedbackActivity |

**Widgets:**

|  |  |
| --- | --- |
| * ImageView * RadioGroup: * SeekBar: * RatingBar * TextView | * Button * **EditText** * RecyclerView * CardView * Fragment * Toast |

**3. Implementation**

* Create database and database connection
* Code the interaction logic for adding and updating participants and payments

A screenshot of a computer

Description automatically generated

Figure 11. Database on Firebase

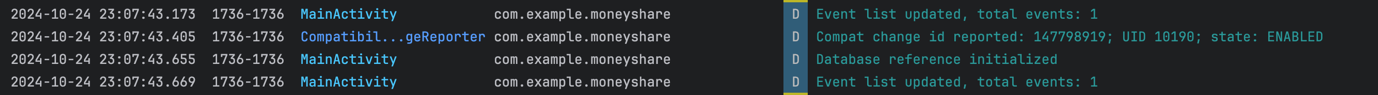
1. **Data Structure**: Firebase Realtime Database is structured to store participants, payments, and events. Each participant has associated payment records and total amounts.
2. **CRUD Operations**: The app implements Create, Read, Update, and Delete operations for participants and payments. For example, adding a participant updates the database and recalculates the amounts owed.
3. **Real-time Updates**: Listeners (ValueEventListener) are set up to automatically update the UI when the database changes, ensuring that users see the most current information.
4. **User Interaction**: User actions (like adding payments) trigger database updates and UI changes, ensuring a seamless experience.

* Handle orientation changes. Implement files in layout-land folder

|  |  |
| --- | --- |
| Figure 12. Portrait orientation | A screenshot of a computer  Description automatically generated  Figure 13. Landscape orientation |

**4: Testing and Debugging:**

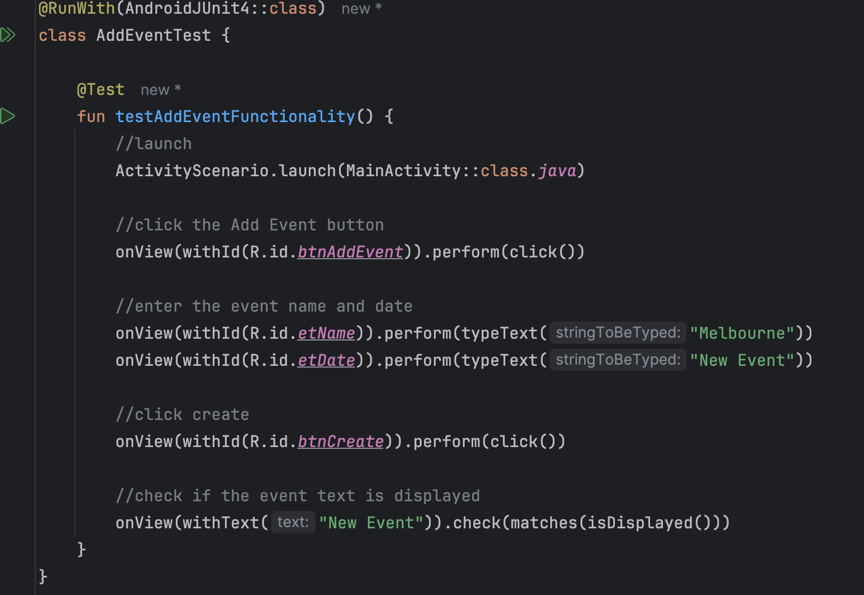
* Test all functionalities on different devices and orientations.
* Log



*Figure 14. Log Demonstration*

* UITesting

Using Expresso for UI Testing



*Figure 15. UITesting*

A black rectangular object with a white line

Description automatically generated

*Figure 16. Test Result*

* Debug.

5. Finalization

* Get feedback from peers or users.
* Submission: Document a report.

6. Future Enhancements (if needed)

**Tools and Resources**

* StackOverflow
* Android Studio
* [ImageColorPicker](https://imagecolorpicker.com/)
* Android Developer
* Chatgpt
* Firebase

**Key Development**

1. **Firebase Advantages**:
   1. **Real-time Updates**: Firebase provides real-time data synchronization, allowing multiple users to see updates immediately without manual refreshes.
   2. **Ease of Use**: Firebase simplifies data management with built-in support for CRUD operations, while Parcelable data (in Assignment2) requires complex serialization and deserialization logic.
   3. **Scalability**: Firebase can handle increased data loads effortlessly, whereas Parcelable is better suited for smaller datasets.
2. **CRUD Implementation**:
3. **Create**: Users can add new events, participants, and payments, storing them in the Firebase database.
4. **Read**: Data is fetched from Firebase to display existing events, participants, and payments in the UI.
5. **Update**: Whenever users add more payments, the application will update the total number of money spent in an event.
6. **Delete**: Users can remove events with changes reflected immediately in the database.
7. **Complexity**:
8. The application combines multiple functionalities (event management, payment tracking, user interaction) and requires handling of various data states.
9. Real-time data synchronization with Firebase adds complexity but enhances user experience by maintaining up-to-date information across devices.