
GitHub Username: dariag00

ExpenseSharer

Description

ExpenseSharer (provisional name) is an app that allows its users to create a shared account for either a single expense or a series of expenses, for example, a trip. The application manages the debts between members and tells its users who has to pay whom and how much.

Intended User

Mainly young people, as they are more likely to have to split expenses and people in general who do group activities sharing expenses.

Features

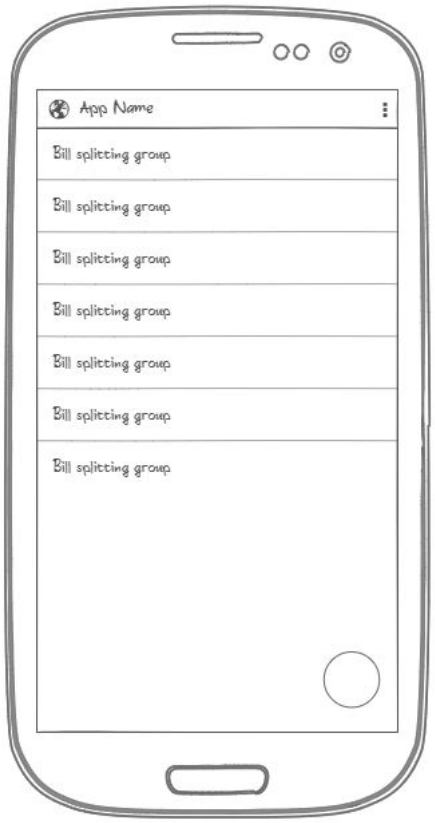
List the main features of the app:

- Create a bill or expense group in which the expenses are divided among the participants
- Manage the groups, adding members, removing members, changing the name or the description, etc.
- Add or remove expenses to one group
- Invite users to join your group.

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



This screen represents the Main Activity and it's in charge of showing the list of groups of the user. At the bottom left, a circular button will allow the user to create or join a group.

Screen 2

This screen contains 3 tabs. The Divider tab is in charge of showing charts about the expenses and of showing how much money each member has to receive/pay. The Expenses tab shows a list of the expenses added to this group. Each expense contains information about who paid it, which amount, a title, etc. These two tabs have a floating button to add an expense to the group. Finally, the Members tab shows the members of the group. It contains a floating button to add a new member to the group.

Add as many screens as you need to portray your app's UI flow.

Screen 3

This screen is the group creation screen. Here, the user will add the details of the group such as the title, the description, the currency used and its participants. There is 2 types of participants:

- Participants without an associated user: These participants are “mock” participants used to represent participants of the group that does not have the app installed
- Participants with an associated user: These participants have to join the group via an invitation code.

Key Considerations

How will your app handle data persistence?

There will be two types of Data Persistence. For the registered users of the app, Firebase Auth will be used. For the rest of the data, the app will use Firebase Realtime Database.

Describe any edge or corner cases in the UX.

No content: When a user is landing for the first time on the group detail page he could find an empty activity with no data at all. The most helpful thing to do could be to add some kind of a textview indicating the user that he/she has to add a payment in order to advance.

Users can upload a picture to their profile but it is optional. Instead of showing an empty hole and no image at all the app could use a default avatar.

Describe any libraries you'll be using and share your reasoning for including them.

The following libraries will be used:

- Butterknife: it greatly facilitates the process of binding the views with their java objects.
- Dagger 2: Dagger 2 provides great dependencies capabilities, such as, autowiring and injection. Dagger 2 analyzes the app dependencies and helps the developer to wire them together. As the app will use multiple data repos, it will greatly facilitate this task.
- MPAndroidChart (https://github.com/PhilJay/MPAndroidChart): A fancy char library to help showing data in a easier way to the user
- Espresso: Espresso is one of the best libraries for UI testing and helps the developer to write concise and reliable Android UI Tests.

Describe how you will implement Google Play Services or other external services.

The app will use Google Play Service with Firebase. More specifically, Firebase Realtime Database and Firebase Auth will be used.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

You may want to list the subtasks. For example:

- Configure and add the libraries
- Create the structure of views and classes following dagger pattern
- Set up the repository pattern for the data persistence

Task 2: Implement UI for Each Activity and Fragment

This task consists of the implementation of the main UI elements. The UI will consist of 3 Activities: MainActivity, DetailActivity and LoginActivity.

List the subtasks:

- Build UI for MainActivity
- Build UI for DetailActivity.
- Build UI for the LoginActivity
- Build Fragment for expenses list
- Build Fragment for expense division
- Build Fragment for members list

Task 3: Implementation of the user system

This task consists of the implementation of the user system. This means implementing the Firebase Auth database and the login system.
Describe the next task. List the subtasks. For example:

- Set up the Firebase Auth database
- Implement the user system in the application

Task 4: Implementation of the business logic of the Expense Divider and the Expense Group

Implement the base business logic of the application. The user could add, delete or modify an expense. When doing so, the expense divider algorithm will take place. This algorithm is in charge of identifying how much money each user has to pay/receive and to whom.

List the subtasks.

- Create the structure of classes
- Create Unit Tests to test the functionality
- Implement the algorithms

Task 5: Implementation of the sharing process

Any user could share the expense group with her/his friends. To do so, he/she will have to click a share button and a code will appear. He will have to share that code with his/her friends.

List the subtasks

- Create layout
- Something else

Task 6: Implementation of the notification system.

The users will be notified whenever a change in their expense group occurs: a new user is added, a new expense is added, an expense is removed, etc.

List the subtasks:

- Implement the notification system
- Add notifications for each action

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"