How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: Songroid

Expense tracker app

Description

Problem:
When checkout in the cash only restaurants, people often find it inconvenient to type all the details or keep receipts for monthly expense tracking. Also, they need options of syncing this data somewhere, like Firebase.

Proposed Solution:
Design an app that allows a user to create cash expense details and sync. For an expense detail containing date, expense and place, the app should let user:

- pick a date from a calendar dialog
- choose a place from a list if it's visited before
- type expense using numeric keyboard as default

After that, the app should provide options to share the data somewhere, preferably Google Keep. Locally the data should be stored in Content Provider. Remotely the data should be synced to Firebase. It's nice to communicate with other Android apps. For instance, the app can launch Google Maps externally if the user needs more info of the place.

Intended User

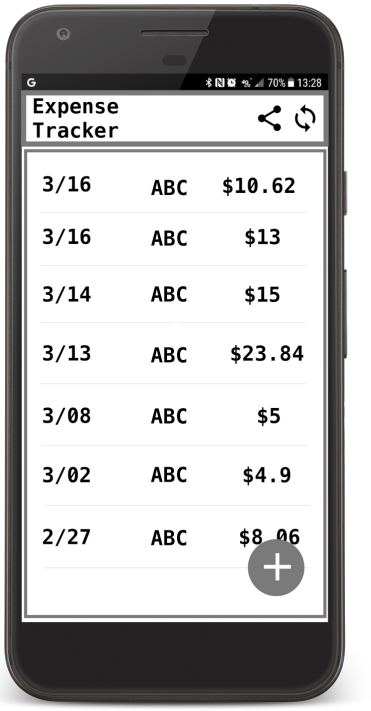
People who usually dine out and track monthly expenses.

Features

- Track transaction details (date, retailer name, expense).
- If the retailer is visited before, next time user can select instead of typing.
- Sync data to Firebase.
- Share data to other productivity apps like Google Keep.
- Show retailer details and auto complete from Google Places API.
- Remove transaction item.
- Multi-select items to remove or share.

User Interface Mocks

Screen 1

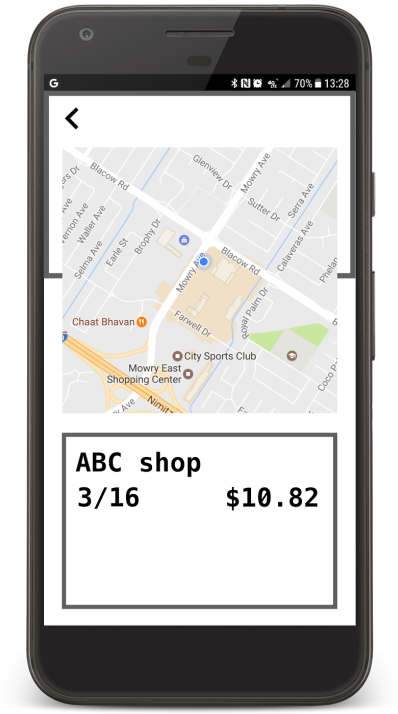


Main fragment
This is the main screen of the app. It contains a list showing the expense, date and place. Pressing each item leads to the detail screen. The toolbar provides sync and share functionalities, which enable sharing rich content to apps like Google Keep and syncing data to Firebase. The FAB button is for adding a new expense. Long press the item selects a specific item, providing removing functionality.

Screen 2

New expense fragment
When FAB is pressed, a new expense fragment is popped up from the bottom. This fragment allows user to select date, type expense and choose a place. My initial thought is to implement place history within a dropdown of the edittext. Maybe providing an integration of Google Map API for auto-complete is a better idea, which depends on how the API looks.

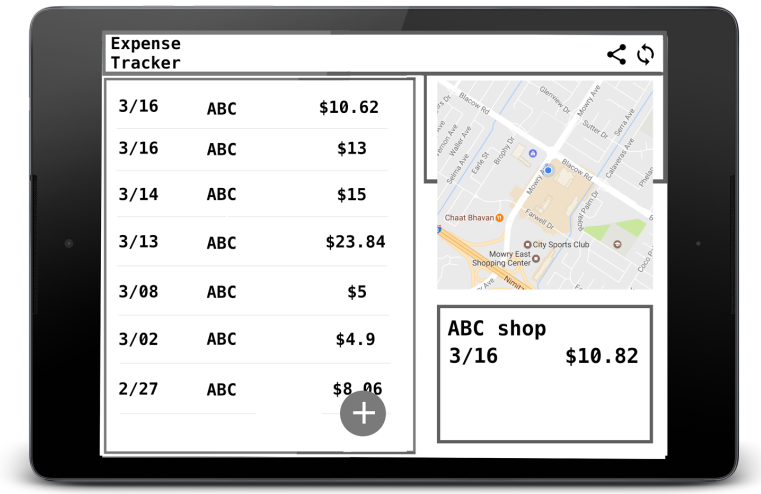
Screen 3



Detail fragment

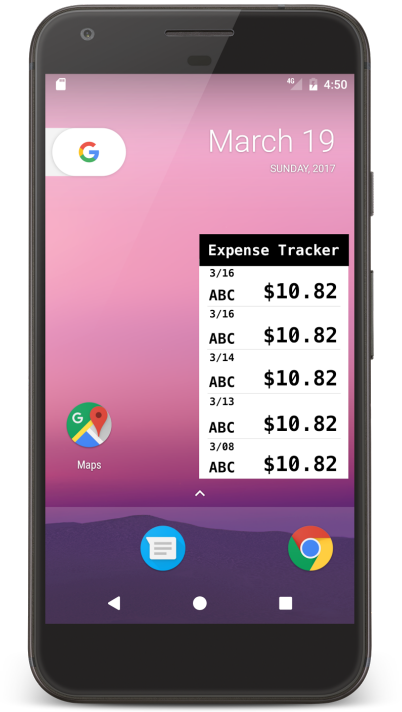
This is a simple detail screen showing additional map information. It's nice to have a list all expenses under the same merchant. However currently I'd like to pass the information as arguments to this fragment (as simple as possible). Also, navigation to this page from the main screen should have shared element transitions.

Screen 4



Tablet
This is the main screen for the tablet. Note that there's no screen transition. The default detail screen's data is from the first row of the list.

Screen 5



Widget screen

Key Considerations

How will your app handle data persistence?

Build a Content Provider.

Describe any corner cases in the UX.

- When new expense fragment is shown, both buttons from the toolbar should function well without any block.
- Pressing the back button should go back to the main screen.
- The place edittext should provide both auto-complete and history options. Will find out a better solution.

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

- Support Library: for material components.

- Okhttp3: for any network calls. We probably don't need to parse the response so okhttp is fine.
- Design Support Library: for FAB, recyclerview and cardview.
- Firebase Assistant: I've never used Firebase before. Not sure the name of the individual libraries. Will follow the tutorial.
- Google Play services: for Google Places API.

Describe how you will implement Google Play Services.

- Google Places API:
  <meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="YOUR_API_KEY"/>

- Google Map preview:
  Follow the instruction of the tutorial and integrate google_maps_api.xml to my project.
- Firebase:
  Follow Firebase Assistant.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Download Android SDK, setup Android Studio.
- Configure all the required libraries.
- Register required keys from Google.

Task 2: Implement UI for Each Activity and Fragment

- Recyclerview, toolbar and FAB for the main fragment.
- For the recyclerview, decide how data is updated.
- Decide which UI component should be used for the popup bottom fragment.
- Recyclerview viewholder.
- Integrate two fragments together for the tablet.

Task 3: Follow up

- Implement Google Play Services.
- Implement loader.
- Handle Error Cases, like empty views and snackbars.
- Handle rotation scenarios.
- Handle offline scenarios.
- Handle permissions.
- Decide how local data is updated when the refresh button is pressed.
- Support for accessibility.
- Add a widget for the most recent 3 expenses.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"