How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: sbaltazar

PemuCoffee

Description

PemuCoffee is an app that brings delicious recipes for coffee preparations. It allows users to view and search recipes by type of coffee drinks (Cappuccino, Americano, Latte macchiato, etc.) or by brewing methods (Chemex, Moka pot, Aeropress, French press, etc). If you like a recipe that the app doesn't have, you can make your own recipe and save it for later use. Search for nearest coffee shops to enjoy your favorite coffee.

Intended User

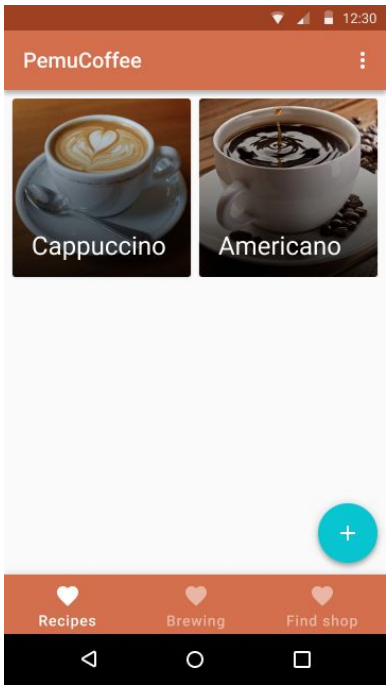
Intended user is everyone who loves coffee (students, programmers, office workers, etc) and everyone that has the desire to learn coffee brewing for personal experience.

Features

- View coffee drinks recipes (View information)
- Share recipes (Share function)
- Make own recipes (Save recipe information)
- View brewing methods
- Show nearby coffee shops (Uses current user location)
- Connect to the internet to get recipes.

User Interface Mocks

Screen 1

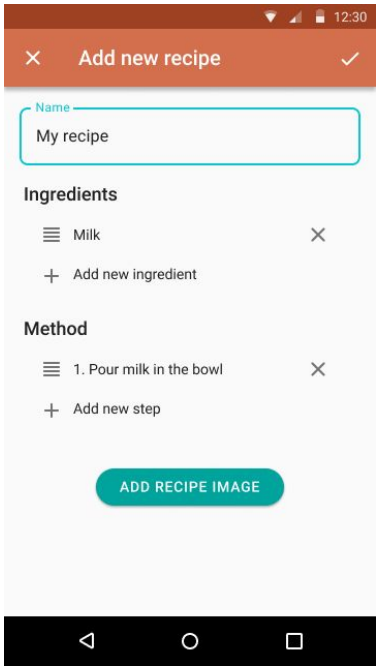


Recipe screen for recipes viewing, with a bottom navigation bar for menu navigation, a button for adding a custom new recipe. Also an option menu for sorting options.

Screen 2

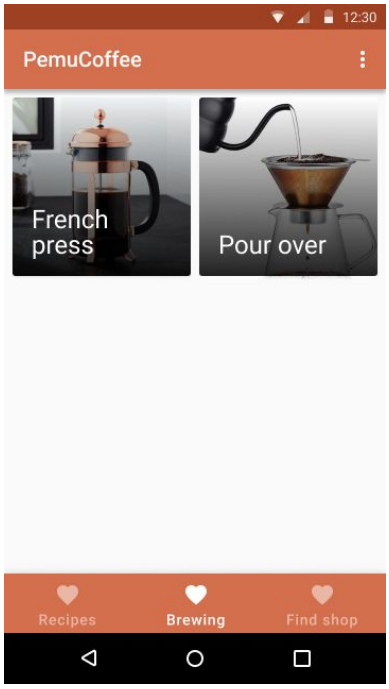
Recipe detail view that shows the coffee drink recipe, ingredients and methods. You can also share the recipe with the share button.

Screen 3



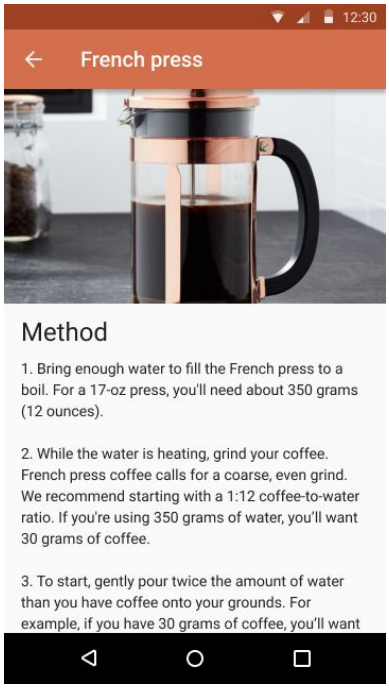
New recipe screen. You can add a new recipe with its ingredients and methods. You can also add an image, by selecting from the gallery or taking a photo.

Screen 4



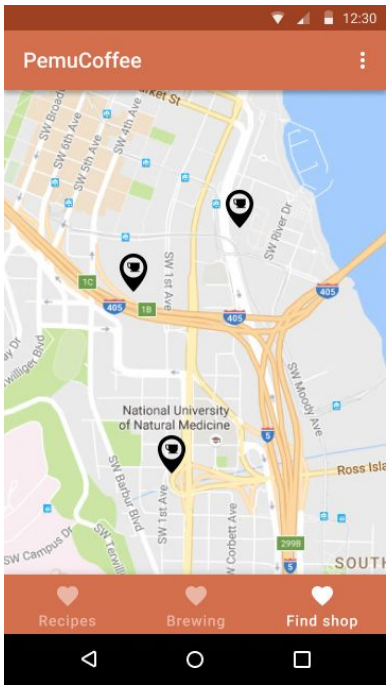
Brew methods screen. Shows the available brewing method for coffee preparations.

Screen 5



Brewing details screen. Shows the brewing method details for preparation.

Screen 6



Coffee map screen. Using user location shows the nearest coffee shops in the area.

Key Considerations

How will your app handle data persistence?

Room ORM will be used for data persistence. For retrieving initial data an API endpoint (An own azure function) will be consumed using Retrofit.

Describe any edge or corner cases in the UX.

The navigation for the app is described below:

- Initial start will retrieve the coffee recipes and brewing methods from the API. If no internet connection is available. The user will be redirected to internet settings. If no internet connection is provided, the app will show a message like “No recipes found. Connect to the internet to view them”.
- For the new recipe screen, when the user wants to add an image, the app will show the permissions dialog required (select image from gallery, take a photo). If no permission is granted, no photo will be attached to the new recipe.
- On the coffee map screen, the user will be asked if he wants the app to use his location. If no permission is granted, the map will be disabled with the option for granting the permission to use this feature.

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

- Glide for image loading
- ~~RxJava for asynchronous operations on the Room database.~~ (ViewModel and LiveData will be used)
- Retrofit for API endpoints calls.
- Timber for logging details.
- Gson for JSON serialization and deserialization

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

- Google Maps API for maps retrieving and locations features.
- Google Places API for getting coffee shops information and details based on user location.
- Azure Functions for initial information to retrieve.
- Google AdMob for revenue (only in the free version of the app.)

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

- Select and build the gradle file for project initialization
- Set the correspondent API keys for Google APIs in the described directory or file

Task 2: Implement UI for Each Activity and Fragment

- Build UI for CoffeeRecipeFragment
- Build UI for CoffeeRecipeDetailActivity
- Build UI for AddRecipeActivity

- Build UI for BrewMethodFragment
- Build UI for BrewMethodDetailActivity

- Build UI for FindCoffeeShopFragment

Task 3: Implement Azure Function

An azure function will be created for providing the information needed for API consumption. The function will return a JSON response.

Task 4: Consuming API Endpoint

Using Retrofit the app will consume the API Endpoint

- Create a class for service related methods
- Define API urls endpoint using the required annotations from Retrofit
- Create API POJO classes for deserialization using Gson
- Call the Retrofit interface for retrieving the data using an AsyncTask

Task 5: Building the database

Room ORM will be implemented for database storage.

- Create the database class using Room annotations
- Create DAOs and Model objects for managing the data
- Create LiveData and ViewModel classes for data retrieving and insert.

Task 6: Implementing Google APIs

Google Maps API and Places API will be implemented for location related functions. AdMob will be used for revenue

- Create required key on Google Cloud Platform
- Store the keys in the project on a file not tracked by Git for security reasons.
- Create helper classes for map initialization to be consumed by the correspondent activity.

Task 6: Build variants

Two variants will be created: free and paid versions. The paid version will not contain any ads.

- Add free and paid flavor to the gradle file. Add the ads dependency only in the free flavor.
- Create helper class for ad initializing, this class will only be available on the free version.
- Update Manifest and provide the key created on the previous task.

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**"