

CocktAPP

Report of mobile app design

Course: SWMAD-01 Mobile Application Development

Study Programme: Software Engineering

Semester: E23

Participants:

Ramon Bendinger 202300832

Aleksander Chotecki 202300831

Tobias Meyhoefer 202300833

Natalia Potrykus 202300834

Emre Temel 202300830

Yohan Unver 202300838

Cesar Vela Martinez 202300826

December 11, 2023

Contents

1 App vision . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 1.1 App goals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 1.2 Personal goals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2 Context . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 2.1 Users . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 2.2 App goals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

3 Features overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 3.1 User stories for feature designing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 3.2 Features design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

4 System architecture and technologies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3 4.1 Packages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3 4.2 UI & UX . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3 4.3 System architecture highlights . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

5 Design overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5 5.1 Use-case diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5 5.2 Navigation diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5 5.3 Graphic design of app views . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

6 Project Tasks and Responsibilities . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

7 Bugs & problems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10 7.1 Bugs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10 7.2 Problems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

8 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10 9 References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

1

1 App vision

1.1 App goals

CocktAPP is meant to be used by adult people. It provides the following options:

• Easy access to numerous fancy cocktail recipes.

• Possibility to add and save user’s own compositions.

• Sharing cocktail recipe via different media.

• Checking the location of open bars in user’s proximity.

1.2 Personal goals

Building this app is a perfect opportunity for all team members to:

• Get familiar with the most modern solutions for mobile application design and development. • Experiment with external API usage.

• Get to know some cocktails classics, so basically get ready for the party!

• Utilize modern architectural patterns to cooperate in a large group efficiently.

2 Context

2.1 Users

CocktAPP user is a person who:

• Is at least 18 years old (considering Danish law).

• Is not a professional barman.

• Owns a smartphone with Android v. 10 or higher with internet access.

• Speaks English.

2.2 App goals

The main objectives of CocktAPP are:

• Providing access to various cocktail recipes.

• Persisting cocktail recipes of users.

• Allowing for easy sharing of cocktail recipes.

• Making it super-easy to find open bars nearby.

3 Features overview

3.1 User stories for feature designing

As a user, I want to:

• be able to save my own recipe so I can access them at any later time.

• mark my cocktails as private or public, so I control access to them.

• browse through many cocktails, not only added by me, so I can find recipes by cocktail name.

• easily share the recipe as text through different communication media, so I can recommend drinks to my friends.

• be able to delete recipes added as my private, so I can have control over my data. • be able to easily find bars next to me, so I can find a place to drink if I don’t fancy doing my own cocktails. • create an account, so I can access my data from different Android devices.

2

3.2 Features design

Using CocktAPP requires registering an account and logging in (using e-mail address and password). The CocktAPP provides three main functionalities for logged in users:

• Search (using searchbar) for cocktails available through connected API (provided the device is connected to the internet), added as ”public” by other users and also saved privately for the current user.

• Add, save and view one’s own cocktails.

• Find the nearest bars in Google Maps.

Moreover, after selecting one of the cocktails (both from retrieved from API and added locally) user can: • View cocktail details.

• Share the recipe as a text via different applications installed on user’s device.

• For private cocktails added by the user - delete the recipe.

All cocktails added by users are persisted in the Firestore [4] database. If user adding the cocktail marked it as ”private”, he will be the only one with access to this cocktail. He will also have an option to delete it. If a cocktail is not marked ”private”, the cocktail will be available to all users of the application and there will be no possibility of deleting it for any user.

Searching for cocktails through ”Search cocktails” screen merges and returns results from two APIs (external and Firebase). The source of the cocktail is indicated in two ways: by a different shade of gray (external/Firebase) and by ”This Cocktail is from:” field in the cocktail row (FirestorePublic/FirestorePrivate/API). There is a depiction in Figure 8.

At ”My cocktails” screen only ”private” cocktails of the user are displayed.

Sharing functionality fills automatically message field of numerous e-messaging apps with recipe formatted as text with title, ingredients list and preparation instruction.

4 System architecture and technologies

4.1 Packages

Following packages were used for managing certain functionalities of the CocktAPP:

| Package | Usage |
| --- | --- |
| Retrofit [1] | HTTP client for communication with external API [2] |
| Hilt and dagger [3] | Dependency injection and code simplification |
| Firestore | Storing user login data and user cocktail data |

4.2 UI & UX

To navigate between different application screens, an approach utilizing Navigation Controller was used. It seemed to be a good choice for a number of reasons:

• It allows for a clear presentation of navigation graph in the app, which makes it easier to develop and extend.

• Transitions between screens look smoother.

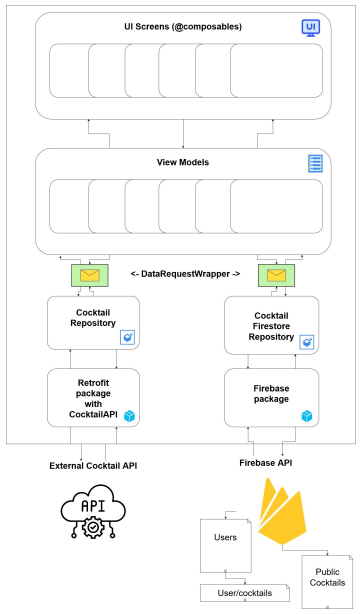
• Passing data between screens is more convenient than in the Activity approach.

To keep things consistent, a dedicated UI theme was created and prototyped before the actual implementation for all planned application screens.

3

4.3 System architecture highlights

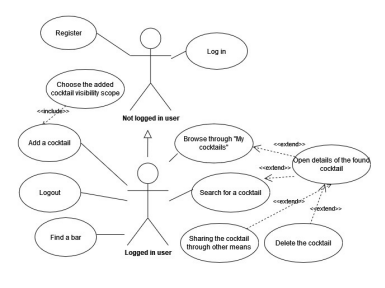
With w team of 7 people, our primary concern was a steady, carefully thought-through system architecture and great separation of concerns, so the team could work as efficiently and in parallel as possible.

Figure 1: System architecture diagram

4

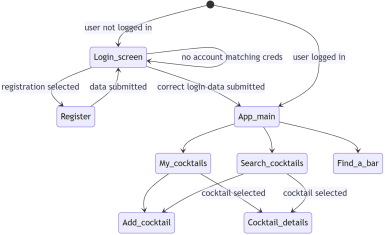
5 Design overview

5.1 Use-case diagram

Figure 2: Use-case diagram for the application

5.2 Navigation diagram

The graph below shows how user can navigate between application features. Main, round-cornered boxes represent app features (screens), while smaller boxes with text over arrows represent conditions required to be met in order to make a navigation step.

Figure 3: Navigation diagram

5

5.3 Graphic design of app views

A modern, minimal and classic design was proposed, utilizing a color theme based on orange and different shades of gray (as shown below). To enhance UX aspects of the application, an animated welcome loading screen was added to greet the user and introduce the app.

Figure 4: Theme colors

To further improve the UX, the user is always made aware of the state off the app. This was achieved by: • feedback messages on user action (such as positive attempt to save a cocktail)

• animated loading sign displayed while fetching data from any API

• displaying a proper info to the user, when any list is empty

Examples of such state indicators are depicted in the figure below.

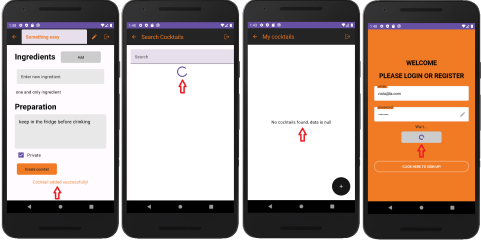
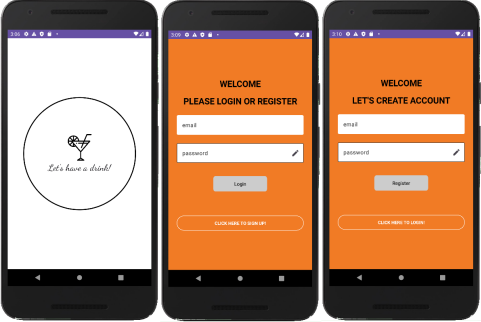
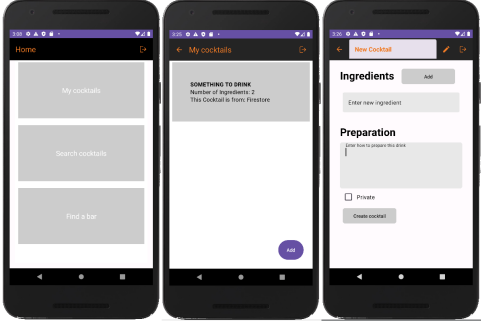


Figure 5: Examples of state indicators in the app, shown as follows: successful cocktail creation, loading cocktail data from APIs, empty list, loading user data from API

The next following 4 figures show different application screens. Figures 6, 7, 8 depict CocktAPP running in the emulator, whereas Figure 9 presents screenshots from the application running on team member phone (to present integration with different applications).

6

Figure 6: Welcome, login and registration view

Figure 7: Main screen, My cocktails view, Add cocktail view

7

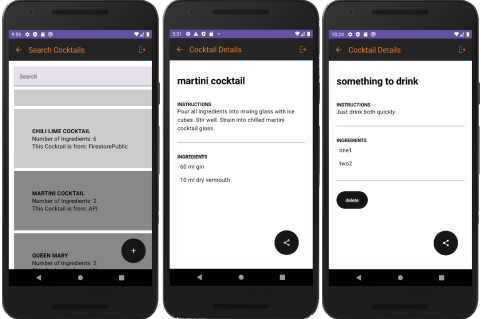
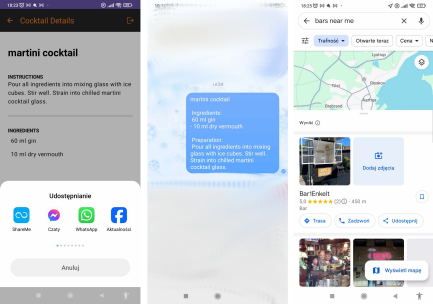


Figure 8: Search cocktails view, Cocktail details view (user public / from external API), Cocktail details view (user private)

Figure 9: Share screen, Formatted recipe (Messenger), Find a bar (Google Maps)

8

6 Project Tasks and Responsibilities

| Task | Who |
| --- | --- |
| Project preparing | |
| Preparing repository & readme | Alex |
| Architecture planning & setting up dependencies | Alex |
| Preparing entry loading screen & adding fonts | Natalia |
| Register and login & Firebase setup & logout | Alex |
| Scaffold with navBar & navController routing | Natalia |
| Connecting external cocktail API | Alex |
| Main screen | Alex |
| Cocktail Row Component | |
| Creating cocktail row in a list (one cocktail representation) | Cesar & Emre |
| Persisting info about which cocktail is retrieved from external API (dark gray), which is from firebase (light gray) | Emre |
| Clicking on a cocktail row redirects to “Cocktail details screen” | Emre |
| Creating a list of cocktails. The same composable for cocktails list is used in both Search cocktails and My cocktails | Emre |
| Search cocktails | |
| Combining results from local and external cocktails | Ramon |
| Showing all available cocktails at first | Ramon |
| Creating search bar (search by partial name from both local/external). API: https://api-ninjas.com/api/cocktail | Ramon |
| My cocktails | |
| Creating a view that shows only own, private cocktails | Cesar |
| Creating “Add cocktail” button that redirects to a proper view | Cesar |
| Add new cocktail screen | |
| Creating a form with entries: name [short text], ingredients (user may add a number of those fields) [short text], preparation [long text], private [bool] | Yohan |
| Saving a cocktail to Firebase collection | Yohan |
| Cocktail details screen | |
| Printing details of the cocktail (name, ingredients, preparation) | Tobi |
| Creating “Share” button for all cocktails | Tobi |
| Creating “Delete” button available only for own, private cocktails | Tobi |
| Sharing functionality | |
| When “Share” is clicked, a new intent should be called, allowing sharing the text version of the recipe through various apps on the user’s phone that allow text sharing | Ramon |
| Formatting the text version of the recipe & filling the “text” field of the external sharing app automatically | Ramon |
| Finding a bar functionality | |
| Opening Google Maps activity (app) to search for bars nearby | Tobi |
| GUI | |
| Layout & design | Natalia & Tobi |
| Styling the app (themes and UI composition) | Tobi |
| Docs & formal | |
| Requirements specification | Alex & Natalia |
| Synopsis | Natalia |
| Final report | Natalia |

9

7 Bugs & problems

7.1 Bugs

Any bugs found by the team were fixed either by members responsible for the faulty part or by the bug spotters. No reported bugs were left unsolved in the project - however, it definitely does not mean that there are none.

7.2 Problems

This app is greatly dependent on availability of external cocktails API. In production this would be a huge risk mitigated by - at least - a regular backup of any external sources.

8 Conclusions

The team managed to create a functional, visually coherent application. There was a significant effort put into the phase of requirements specification, system architecture planning and technology choice - but it turned out to have marvellous impact on the possibility to work in parallel on numerous application functionalities and on the ease of integrating them together.

9 References

References

[1] Retrofit package: https://square.github.io/retrofit/

[2] External cocktail API: https://api-ninjas.com/api/cocktail

[3] Hilt & Dagger: https://developer.android.com/training/dependency-injection/hilt-android#hilt-and-dagger [4] Firestore: https://firebase.google.com/docs/firestore

10