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: Protino

Codewatch

Description

Codewatch is a productivity app for programmers or developers in general. It is an unofficial android client of Wakatime.com which tracks coding activity by providing dozens of plugins for almost all the IDE's out there. Codewatch is an extension which allows users to set goals and deadlines for each project or programming language they are working on.

Intended User

Primarily intended for developers. The user must be a member of wakatime.com which helps to track coding activity.

Features

Main features of the app

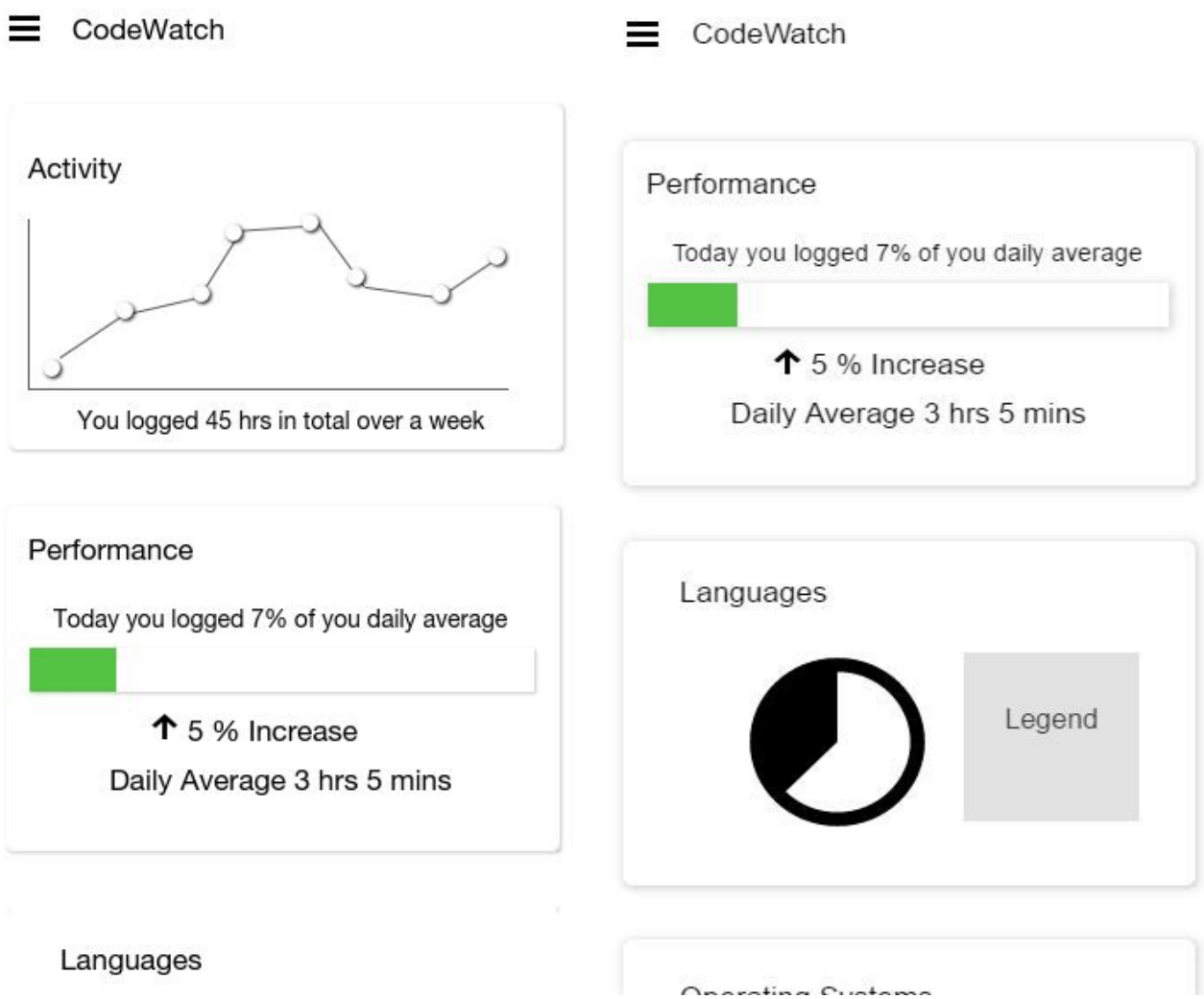
- Track coding activity over the week.
- Setup project and coding goals.
- Add deadlines to projects and track the progress.
- Compare yourself with others through leaderboards.
- Unlock awesome badges after each achievement.

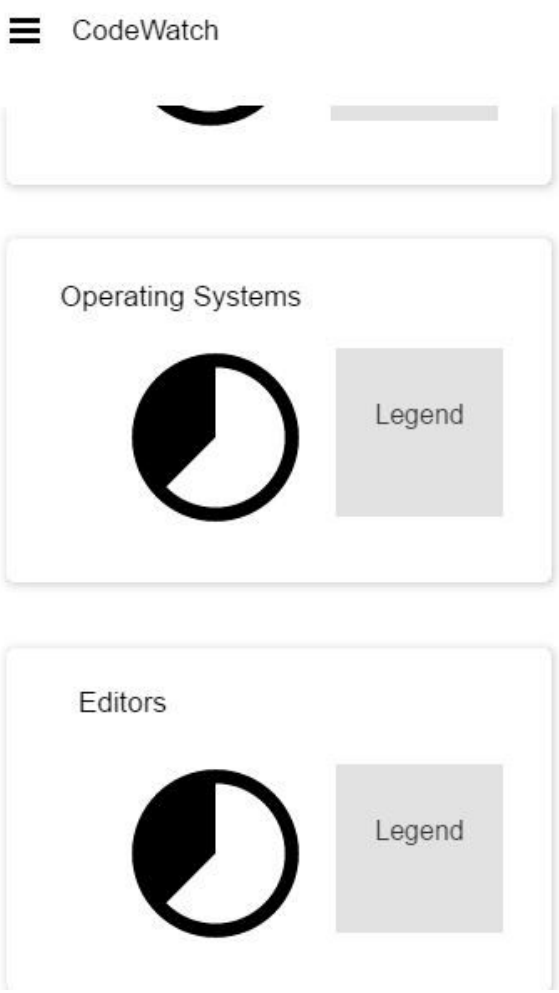
User Interface Mocks

**Note - All the following mocks were created using [proto.io](), you may want to interact with the UI directly*

- *https://pr.to/8GP8FS/*

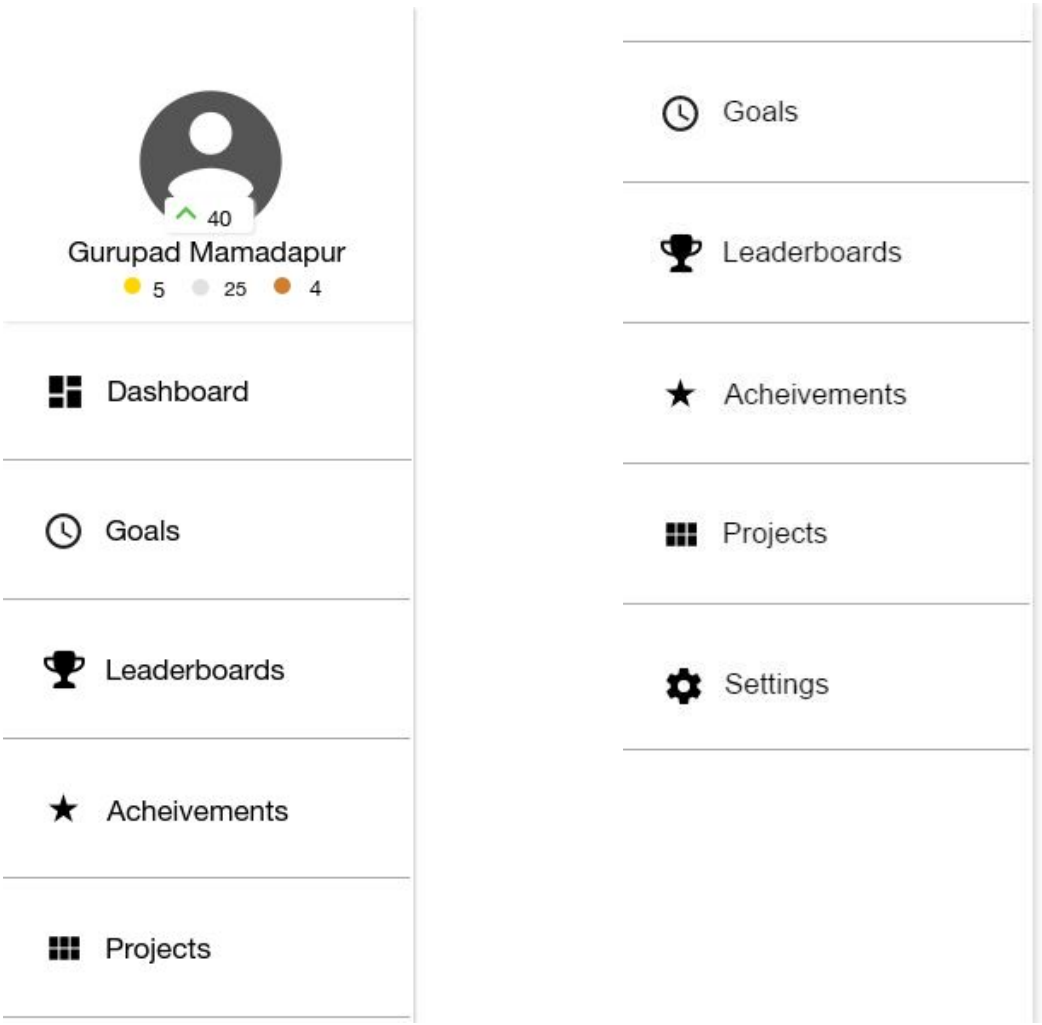
Screen 1





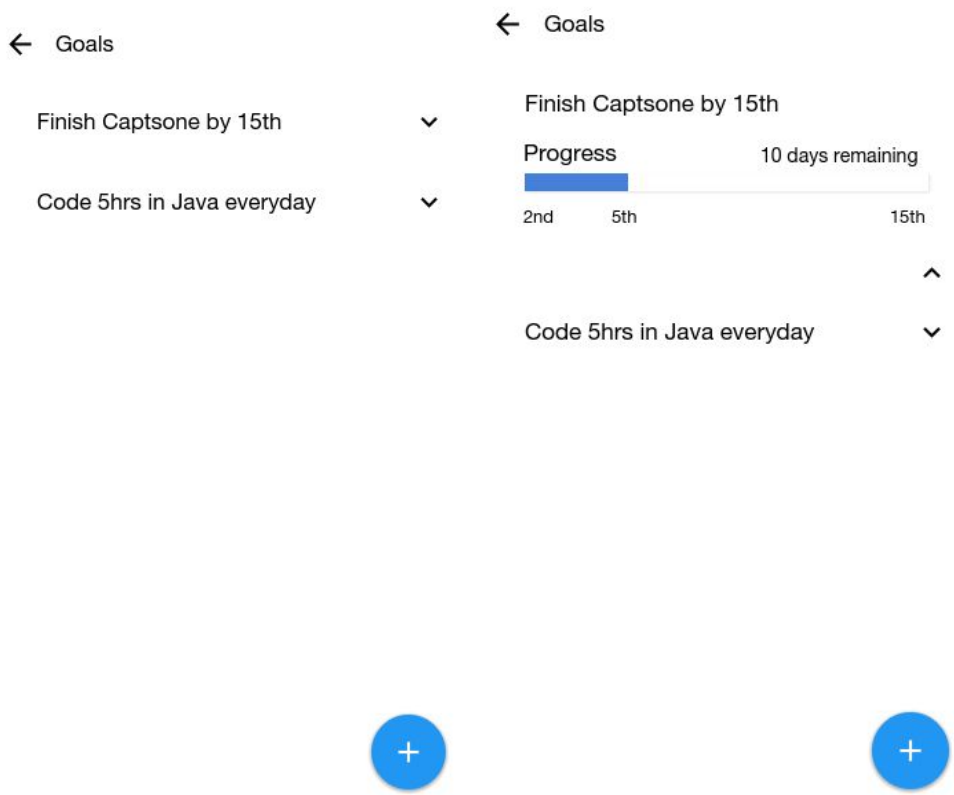
This is the dashboard screen. The entry point to the application after login. This contains All the coding activity and statistics data fetched from Wakatime.

Screen 2

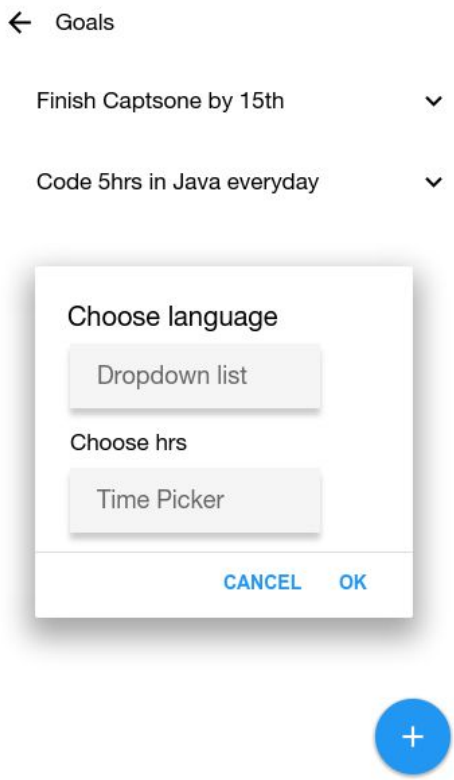


This is the navigation drawer that allows user to browse other parts of the application. It also shows some data related to the user at the top. Clicking on it will lead to profile page

Screen 3

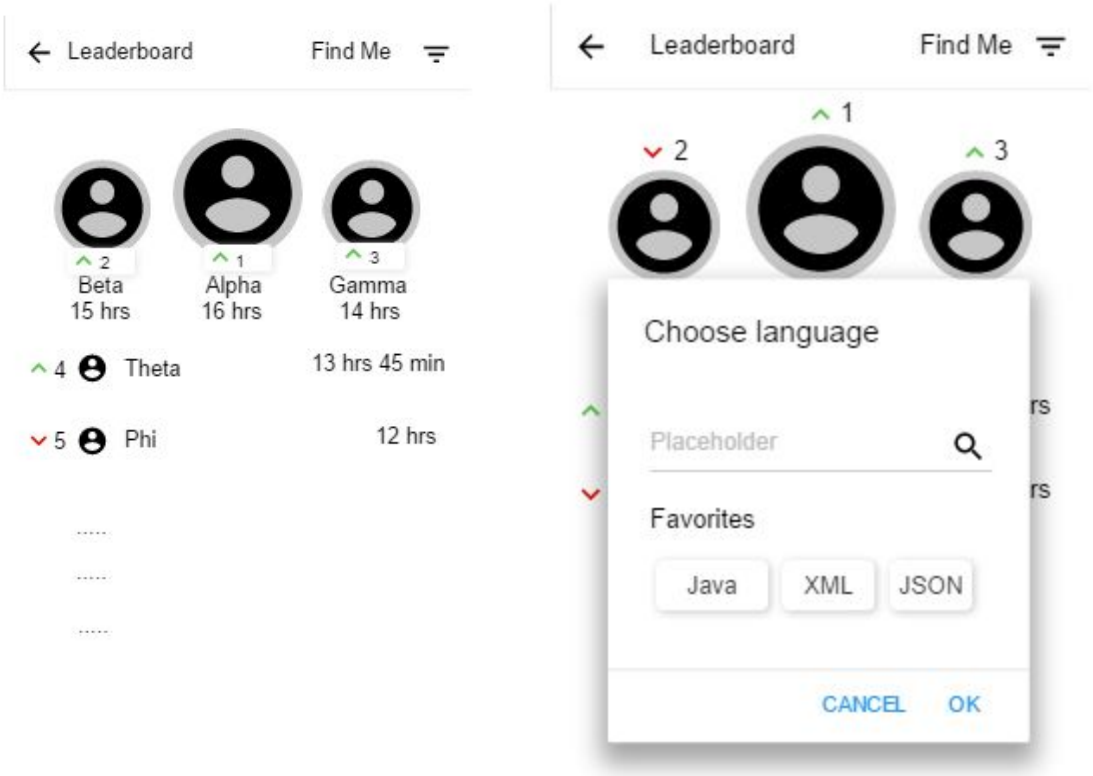

Goals
Finish Captsone by 15th
Code 5hrs in Java everyday
Goals
Finish Captsone by 15th
Progress
10 days remaining
2nd
5th
15th
Code 5hrs in Java everyday


Goals
Finish Captsone by 15th
Code 5hrs in Java everyday
5hrs
Days ->
Goals
Finish Captsone by 15th
Code 5hrs in Java everyday
Choose Goal Type
Finish a project by X date
Spend X hrs on a project daily
Code in X for Y hrs daily
Goals
Finish Captsone by 15th
Code 5hrs in Java everyday
Choose Project
Dropdown list
Choose Date
Date picker
CANCEL
OK
Goals
Finish Captsone by 15th
Code 5hrs in Java everyday
Choose Project
Dropdown list
Choose hrs
Time Picker
CANCEL
OK



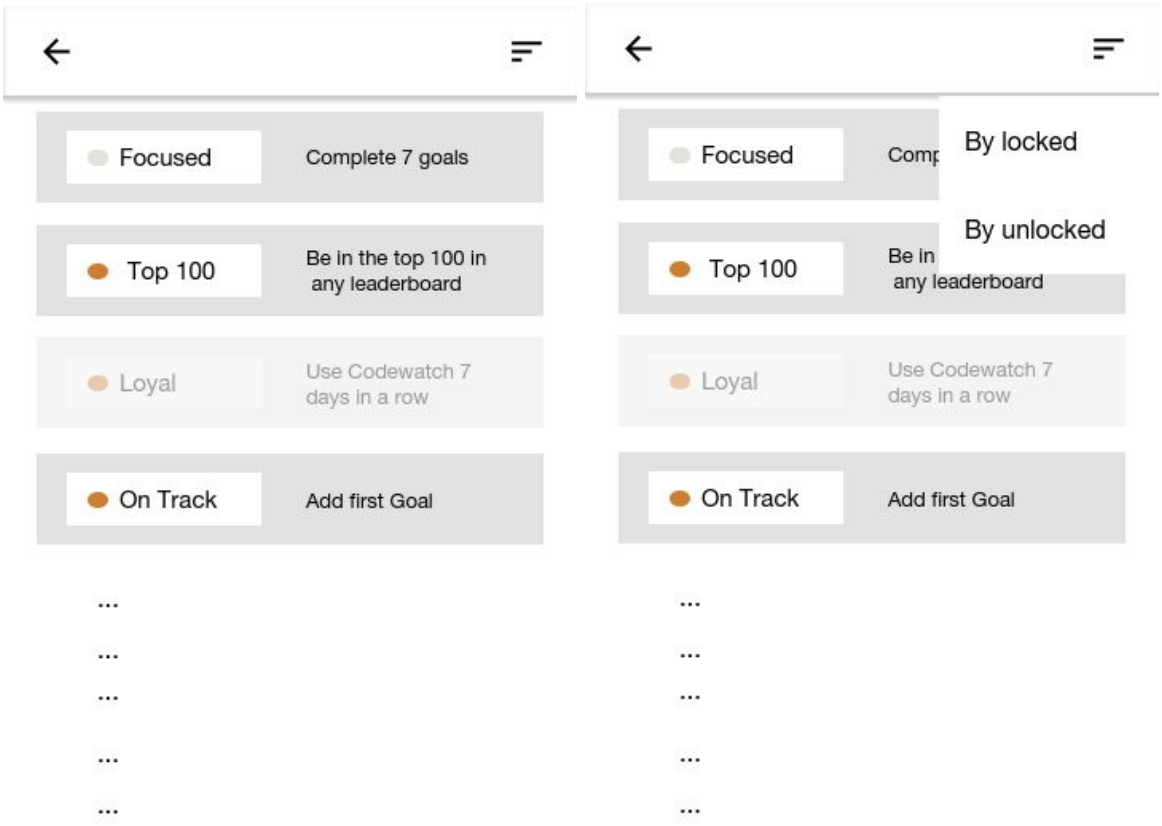
This is related to goals. User can set deadlines, or set hours he/she must everyday in a particular language etc.

Screen 4



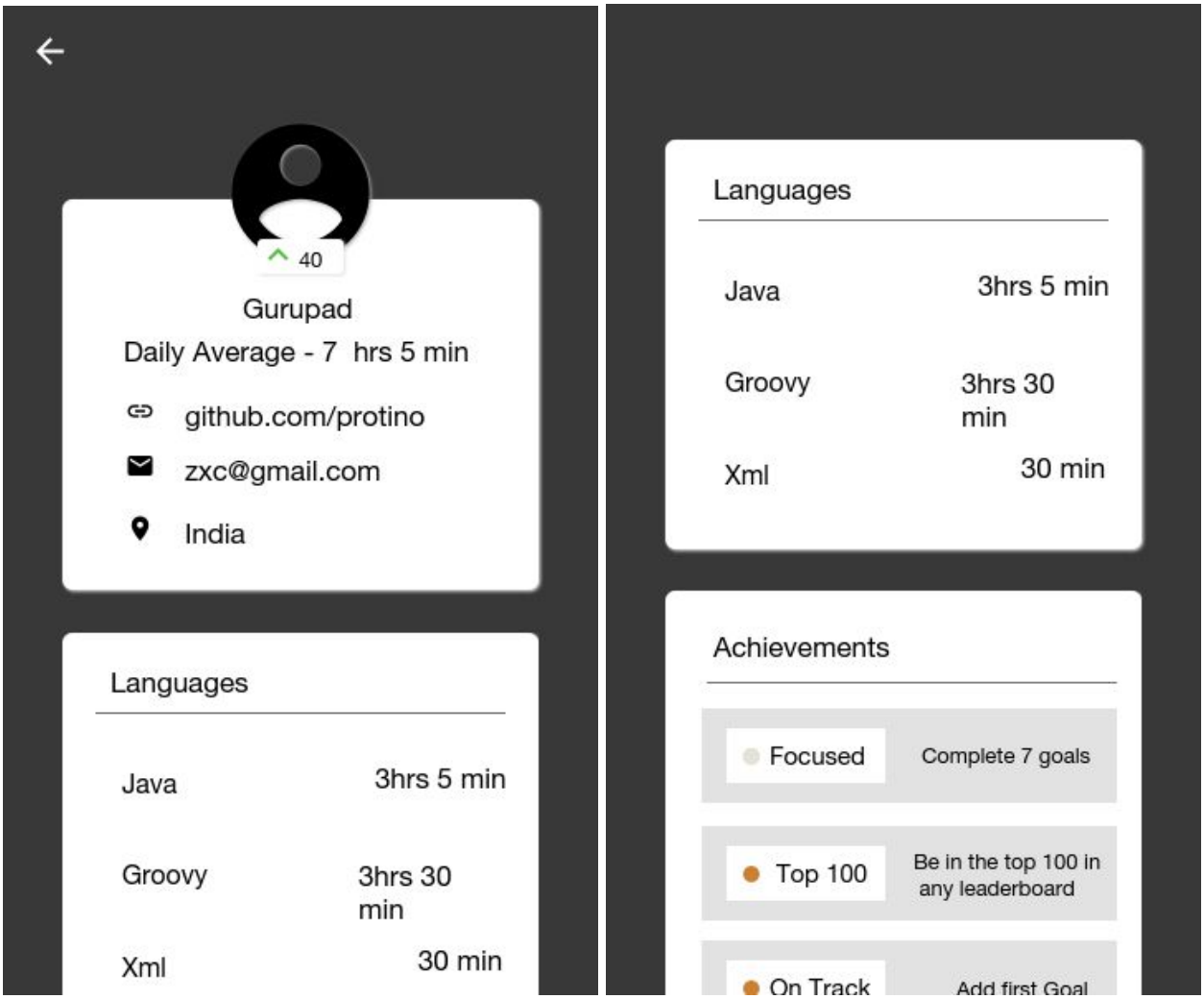
Leaderboards screen, showing who is has highest daily coding average . It also has a filter option to get leaderboard based on language.

Screen 5



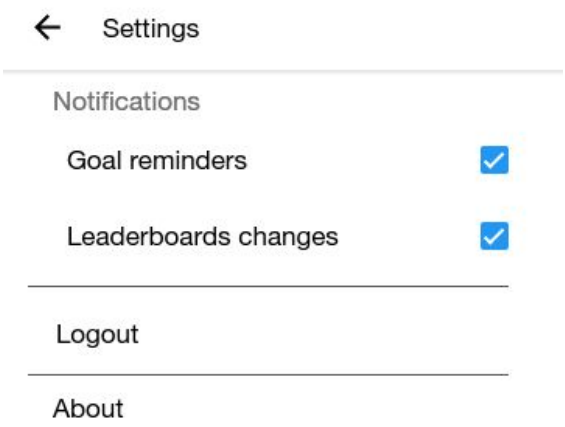
This is achievements screen (I forgot to put the title in toolbar). There are three categories gold, silver and bronze. It's sorted by default in the order - gold, silver, bronze.

Screen 6



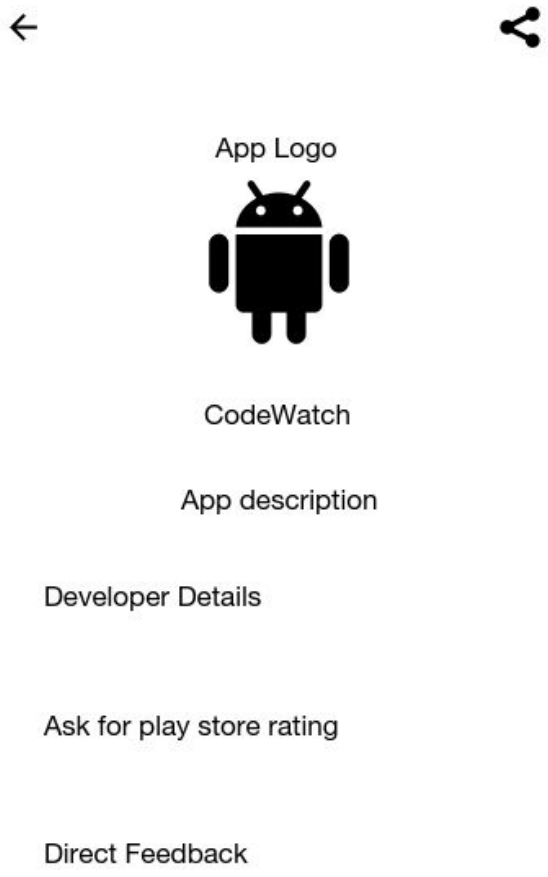
Profile screen. It display user contact, location, rank, other links, coding statistics and achievements.

Screen 7



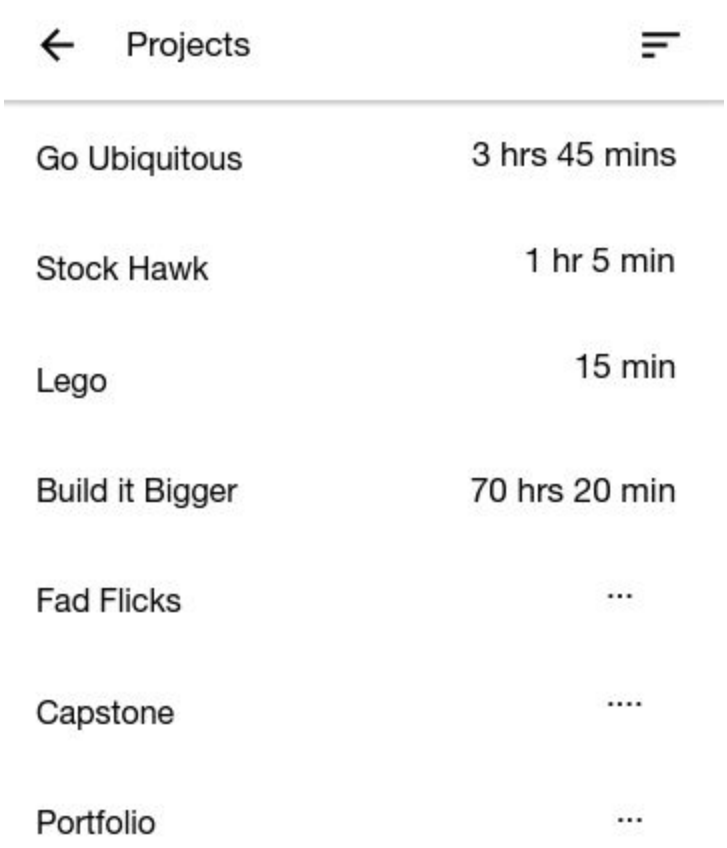
Settings screen that has settings to show reminders or not, logout option and way to open about page.

Screen 8



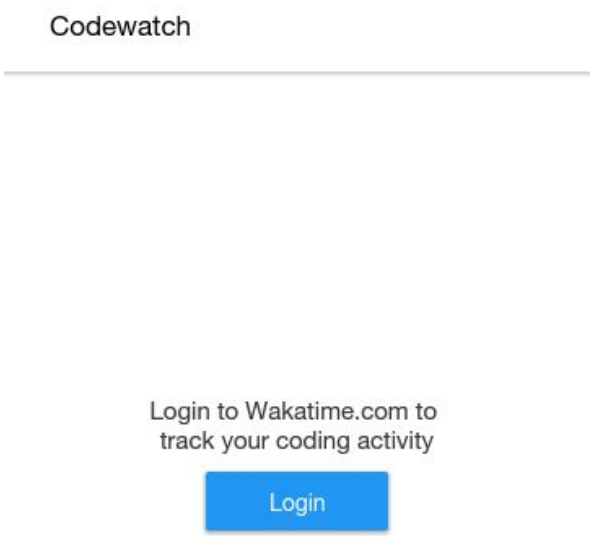
Simple about page.

Screen 9



Here all the projects of users are displayed along with time spent on the project over a week.

Screen 10 - Login



Simple login screen.

Simple widget -

It shows users performance. The bar represents how much is left yet to reach the daily average.

Key Considerations

How will your app handle data persistence?

The data persistence for most of the data is handled by Firebase RealTime Database. Data related to sign-in info, settings and other smaller sized data is handled by sharedPreferences. Leaderboards data is handled by content provider backed by sql database. I chose local sql database for leaderboards because

- Data is slightly large in size so firebase realtime db is not an option.
- Filtering is easy with db queries.
- Also, using a loader to bind data to views is a rubric requirement in stage 2.
- I get to refresh content provider concepts.

Describe any corner cases in the UX.

- After sign-in using Oauth, make sure the browser is closed.
- Sometimes Wakatime responds error message stating - “Calculating statistics for the user, try again Later” Hence display a loading screen or Even better option would be to show old data, if present show a simple snackbar with the above mentioned message.
- When offline, make sure user is notified to connect to the internet via snakbar.

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

Libraries I’ll use -

- MPAndroidChart to draw bar charts and pie charts.
- Picasso to display user profile photo.
- Retrofit2, Okhttp, Gson for Ouath2 sign and fetching data from wakatime api endpoint.
- Icepick to maintain state instance
- Event bus to simplify communication between activities and fragments.
- Butterknife to fetch references to resources.
- LeakCanary to detect memory leakages
- Timber to log debug messages during development.
- Support library to use Material design components.

Describe how you will implement Google Play Services.

I’ll be using the following services provided by Firebase:

- Firebase Realtime Database to store data persistently.
- Firebase Analytics to log events when user uses goals and achievements section of the app. So that I’ll If the user is interested in them or not. Also, to get overall app statistics.
- Firebase RemoteConfig to force users to update to newer version of the app. There can be a scenario where wakatime updates its API endpoint or a major bug fix to reflect.
- Firebase Crash Reporting to make sure I can get rid of bugs as early as possible.
- Admob to monetize the app with simple banner ads.

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: Login Setup

- Setup Oauth with Retrofit 2.
- Make sure the browser closes after login.
- Store access_code in sharedPreferences.

Task 2: Implement all backend related modules

- Fetch all necessary data from wakatime using Retrofit.
- Extract only relevant data and create a POJO class to store user information.
- Also add fields related to goals and achievements to the POJO class.
- Write all the data to the Firebase Realtime database.
- Fetch leaderboards data and store in a SQL database.
- Implement content providers to access leaderboards data
- Add tests to ensure all the data can be written and read correctly.
- Use FirebaseJobDispatcher for firebase and Syncadapter for sql db to update data every day.

Task 3: Create UI

Code all the UI with fake data for now.

- Build Dashboard srceen with navigation drawer. Use MPAndroidChart to display all the charts.
- Build UI for all the other items present in the navigation drawer.
- Build UI for settings screen, about page and login page.
- Preserve instance state after screen rotation.
- Implement UI for tablets and other screen sizes.
- Build widget UI with fake data.
- Follow Material Design guidelines.

Task 4: Attach data to UI components

DIsplay real data.

- Use cursor loaders to display leaderboards.
- Attach listeners to appropriate database references of firebase and load data into other UI components.
- Check for corner cases in UX.
- Make the app functional. Use event bus to unlock badges after a certain task is completed.
- Implement filtering functionality based on languages for leaderboards.
- Add simple animations for performance bars and charts.

Task 5: Implement Firebase features

- Add Firebase analytics to analyse which parts of the app users spend most of their time.
- Add Firebase Crash Reporting.
- Setup RemoteConfig to force users to update the app in case of a major fault.
- Display banner ads on dashboard screen only.
- Configure free and paid build variants.

Task 5: Productionize the app

- Support for localization and addition of accessibility features
- Handles error cases like no internet or server fault etc.
- Document the whole code.
- Create notifications for reminders to finish a goal.

Task 6: Test the application

- Test the app on all types of API versions - minimum to targeted version.
- Test the app on different screen sizes and orientations.
- Test build variants.

Task 7: Final tasks

- Generate signed apk
- Add Apache license
- Write a detailed README

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**"