Assignment #02

IT University of Copenhagen (ITU)
Mobile App Development, BSc (Spring 2019)

Gustav Kofoed Clausen (gucl)
April 29 2019

1 Introduction

This second assignment in the course, Mobile App Development, aims to complete the ninth and final version of the mobile application, *BikeShare*.
The purpose of BikeShare is to enable sharing of bicycles[1] among a community of users in the city of Copenhagen, Denmark. The user is able to register bikes for others to rent, rent bikes themselves, and manage their user account. Furthermore, the app is intended for an environment where the bikes are locked with a special lock that identifies the bikes, and that can be locked and unlocked using Bluetooth. This functionality is however left out of this version, and is replaced with a mock implementation.

This report includes a user guide, a list over test cases used to verify that the functionalities work as intended, as well as a description of the most important design choices regarding the functionalities, the user interface and the overall structure of the app.

2 User guide

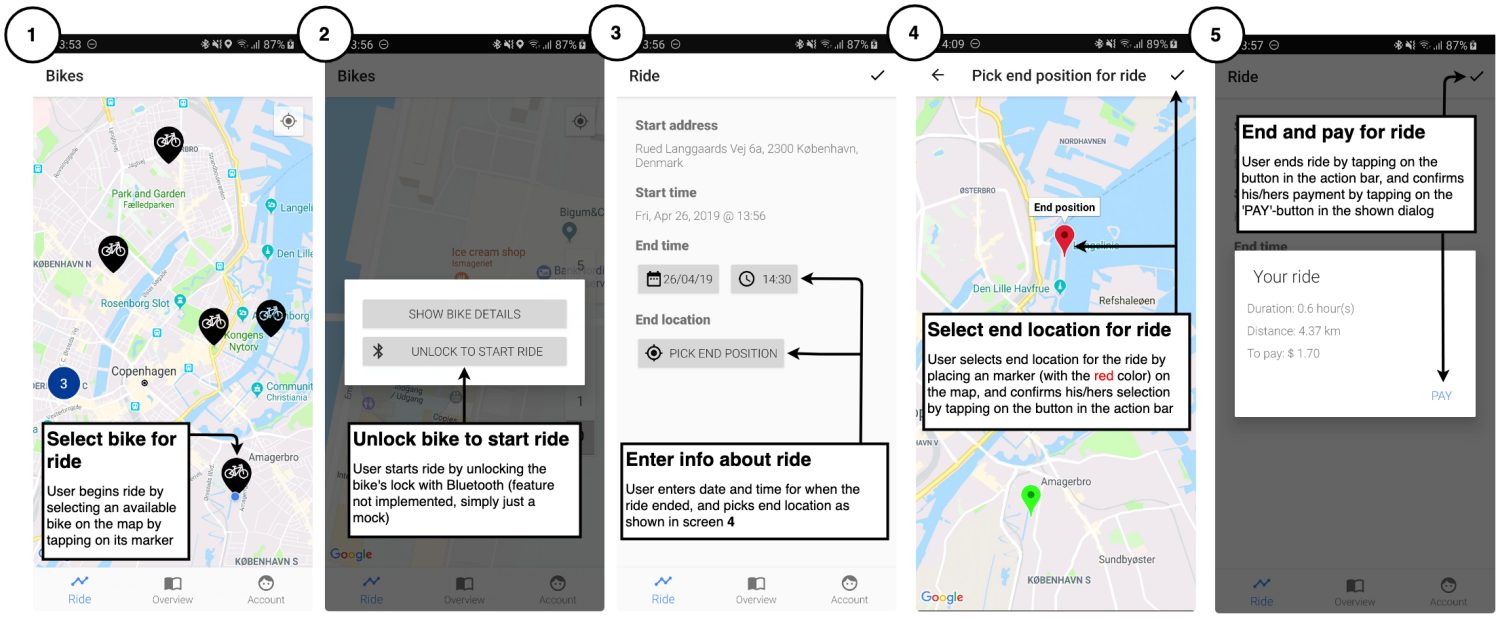


Figure 1: Ride section. Walkthrough of starting and ending a ride.

[1] Hereafter referred to as *bikes*

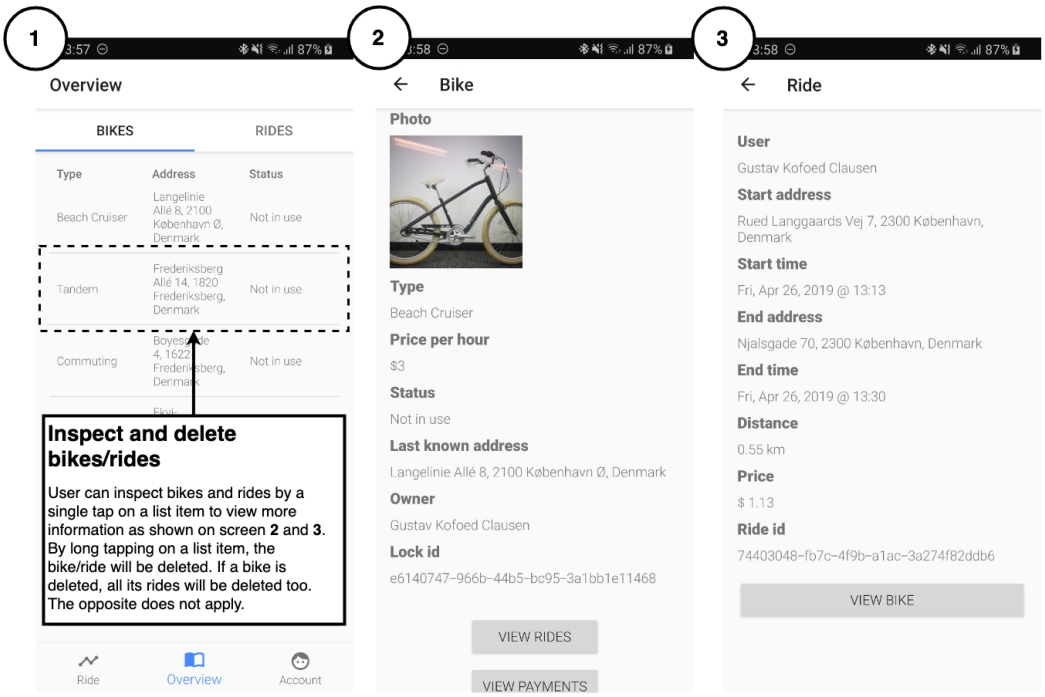


Figure 2: Walkthrough of overview section

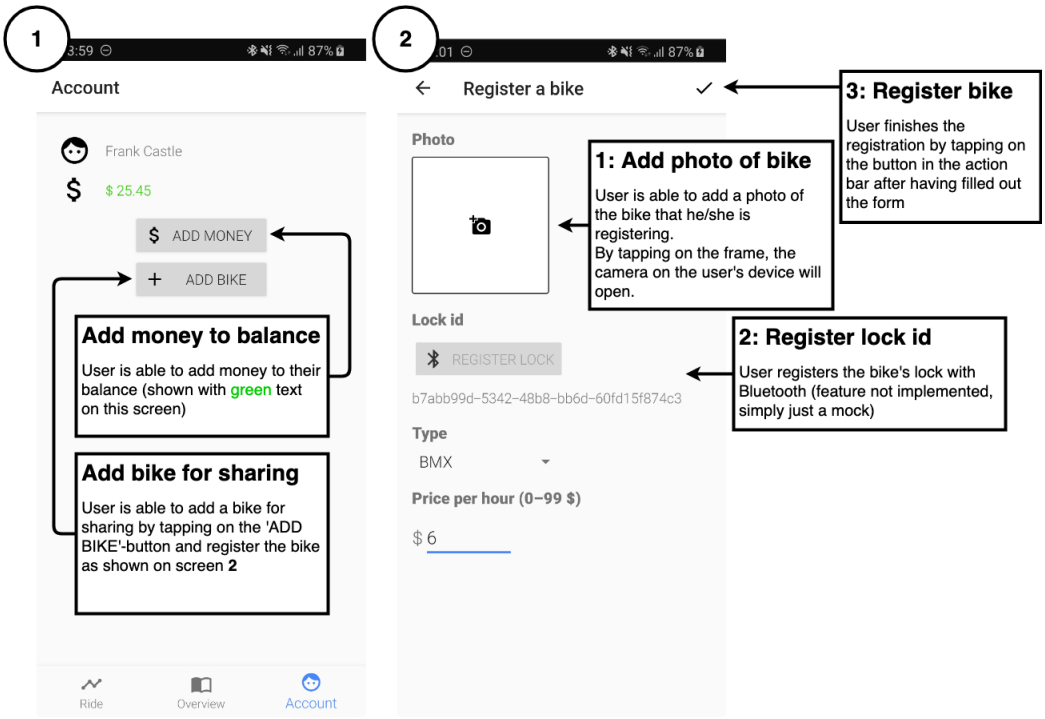


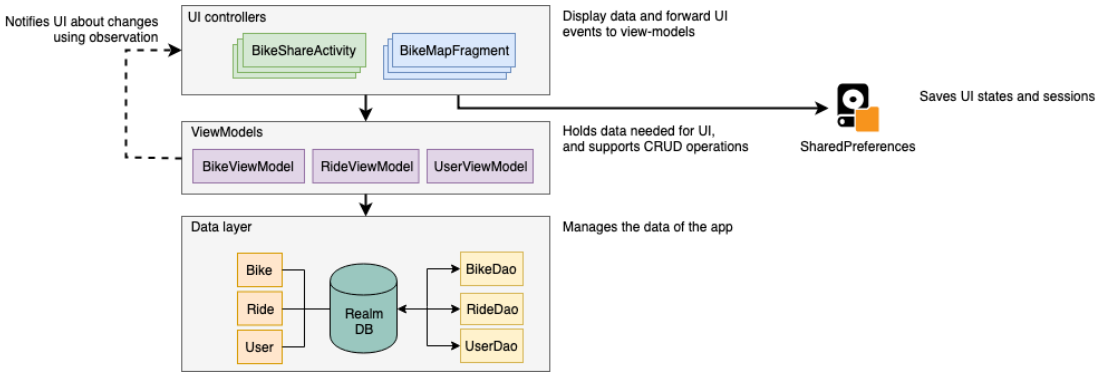
Figure 3: Walkthrough of account section

3 Design choices

Structure

In the diagram below, the general structure of the app is illustrated. This structure is heavily inspired by *Android Architecture Components*[2] that is a simple approach to provide a robust structure to an Android app.

[2] *Android Room with a View - Kotlin*, codelabs.developers.google.com, November 5 2018, `https://codelabs.developers.google.com/codelabs/android-room-with-a-view-kotlin/index.html?index=..%2F..index#0` (accessed 2019-04-26)



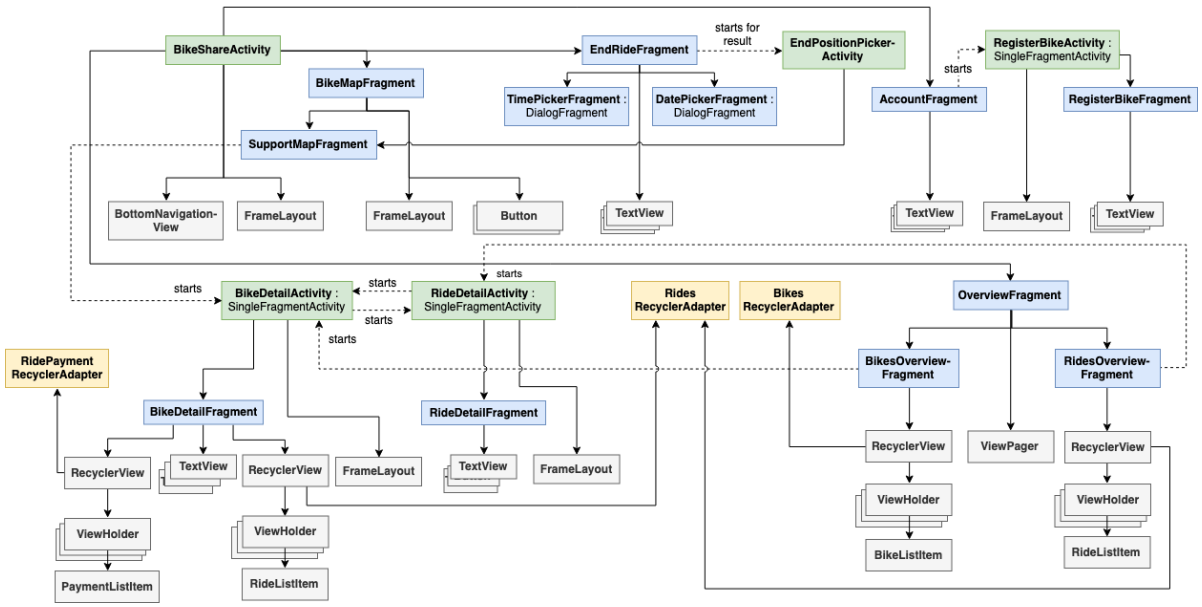
As shown in the diagram, this structure forces a clear separation of concerns for classes that interacts with the user interface (UI) and data. For instance, activities and fragments is only responsible for displaying data and forwarding events to view-models, that is responsible for holding on to UI-related data that is not destroyed on app rotations. Furthermore, they provide the UI a simple CRUD (create/read/update/delete) interface to operate on the data that the UI is displaying. Also, the view-models notify the UI about changes to the data for the UI-components that is set to observe the specific data set.

The view-models operates with data via Daos (data access objects) in the the data layer. For this specific app, *Realm* has been chosen as its database. Realm is an object database which frees the translation to a relational model. Thus, data objects are the data model so it makes it easy to operate on directly.

At last, the UI uses *SharedPreferences* to persist UI states and the user session even after the app is closed. For instance, a reference to the current on-going ride is saved to decide which fragment to inflate in the 'Ride'-section (start or end ride). Also, the current logged in user and the state of the bike map (zoom, position, tilt etc.) is stored using SharedPreferences.

UI

An object diagram illustrating the class and layout structure for the UI is illustrated below. The purpose of this diagram is to give an overview of how the UI is implemented and divided in different activities and fragments.



Functionalities

Three different functionalities in the app has been implemented with an alternative solution to the problem that it tries to solve. Firstly, the distance between the start and the end location of a ride is calculated as a great-circle distance and not by a shortest path algorithm. This has not been implemented due to time constraints. Secondly, as described, the bike pairing with Bluetooth is replaced with a mock implementation. Again, this has not been implemented due to time constraints. Lastly, instead of tracking the device during a ride session, the user is asked to end the ride where they want on a map. This has been done to avoid clustering of bikes in one location while testing, and it is personally seen as a fun feature.

4 Test and evaluation

To verify that the app fulfills the desired functionality and to discover unwanted bugs, a basic test suite with manual system tests has been continuously been updated throughout the development process of the app. This has meant that regression testing could be performed after each increment to constantly verify that the app works as intended in terms of functionality.
Below, five test cases covering most of the app's functionalities is presented.

Test case 1 – Permission handling and accessing required services from device

Step	Result
Open the app for the first time with *Location* and *Network Service (Wi-Fi/Cellular)* disabled on the device, and deny the location permission dialog. Verify alert dialog appears, and app exits with toast after tap on 'OK'-button.	✓
Reopen the app, and tap on 'CANCEL'-button in the alert dialog. Verify app exits with toast.	✓
Reopen the app; tap on the 'OK'-button in the alert dialog; accept the location permission dialog. Verify dialog informing about disabled location service on the device is shown, and app exits with toast after tap on the 'OK'-button.	✓
Enable the location service on the device, and reopen the app. Verify dialog informing about non available internet connection is shown, and app exits with toast after tap on the 'OK'-button.	✓
Enable the network service on the device, and reopen the app. Verify your current location is shown on the map (if located in or around Copenhagen).	✓

Test case 2 – Start ride

Step	Result
Open the 'Ride'-section in portrait mode, and tap on a bike marker on the map. Verify a menu with the option to unlock the bike is shown.	✓
Rotate the device to landscape mode, and verify that the options menu is still open.	✓
Unlock bike. Verify a screen where information about the current ride is shown.	✓
Close the app, kill the process, and reopen the app. Verify that the last screen is still shown after restart.	✓

Test case 3 – End ride

Step	Result
Open 'Ride'-section with a ride started and network service disabled. Verify alert dialog appears, and app exits with toast after tap on 'OK'-button.	✓
Enable network service; reopen app; select date and time before start time; try to end ride by tapping on the button in the action bar. Verify current ride is not ended, and toast displaying error message is showing.	✓
Select date and time after start time, and try to end ride by tapping on the button in the action bar. Verify current ride is not ended, and toast informing about missing end location is showing.	✓
Tap on 'PICK END POSITION'-button. Verify new screen with map is shown, and start position is marked with a green marker on the map.	✓
Try to confirm end position by tapping on the button in the action bar. Verify that user is not navigated away from screen, and toast informing about missing selection of end location is showing.	✓
Pick end position on map and rotate screen. Verify that both start and end position markers is still shown on the map.	✓
Confirm end position by tapping on the button in the action bar. Verify that the nearby address of the end position is shown below the 'PICK END POSITION'-button.	✓
Rotate screen. Verify that both the entered date and time and nearby address persists.	✓
End ride by tapping on the button in the action bar, and rotate screen. Verify that the payment dialog is still showing.	✓
Tap on the 'PAY'-button in the payment dialog. Verify that the borrowed bike is placed at the end position on the bike map; that the ride and its payment can be found in the borrowed bike's detail view; that the ride's price is deducted from the user's balance found in the 'Account'-section.	✓

Test case 4 – Register bike

Step	Result
Disable location and network service, open 'Account'-section, and tap on the 'ADD BIKE'-button. Verify alert dialog informing about disabled location service on the device is shown, and activity closes with toast after tap on the 'OK'-button.	✓
Enable location service, and reopen screen. Verify alert dialog informing about non available internet connection is shown, and activity closes with toast after tap on 'OK'-button.	✓
Enable network service, and reopen screen. Verify that screen with bike registration opens.	✓
Tap on camera frame, and take picture with the device's camera. Verify that the picture shows up in the camera frame in the bike registration screen.	✓
Try to complete registration by tapping on the button in the action bar. Verify that registration does not complete, and toast informing about missing lock id is shown.	✓
Tap on the 'REGISTER LOCK'-button, and try to complete the registration again. Verify that the registration does not complete, and toast informing about missing price shows.	✓
Enter price, and rotate screen. Verify that all the information in the form persists.	✓
Submit form. Verify that the registered bike shows up near your current location on the bike map in the 'Ride'-section, and that the submitted information about the bike is shown in its detail view.	✓

Test case 5 – Delete bike and ride

Step	Result
Open 'Overview'-section, navigate to the ride tab, and long tap on a list item to remove the ride. Verify that the item is removed from the list, and that the ride and its payment is not to be found in its bike's detail view.	✓
Switch to the bike tab, and long tap on a list item to the remove bike. Verify that the item is removed from the list; that the removed bike's rides is removed from the ride list; and that the bike is removed from the map in the 'Ride'-section.	✓
Start a ride with a bike in the 'Ride'-section, and remove that bike in the 'Overview'-section. Verify that the current ride is deleted when navigating back to the 'Ride'-section.	✓

5 Problems

The only issue encountered with the app in its current state is that it does not function properly when it is run on an emulator. Each time that the app is opened, it will close since the default Android emulator does not support location services out of the box. Thus, it will fail the first test case as presented in the previous section. It is therefore required to run the app on a real, physical Android device.