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: skaarberg

BikeFriend

Description

Do you need to find the closest available city bike in Oslo? Do you want to find the closest available city bike parking, or do you want to check the status of your favourite bike racks before going to school or home from work? Let me be your BikeFriend and help guide you there!

Intended User

This app is intended for everyone who wants to use the city bikes in Oslo. Whereas the official app (https://play.google.com/store/apps/details?id=no.urbaninfrastructure.bysykkel) has features to find avaiable locks, BikeFriend provides useful additional help when users does not know how to get to the bike rack, so it is especially useful to people who are not that familiar with the area.

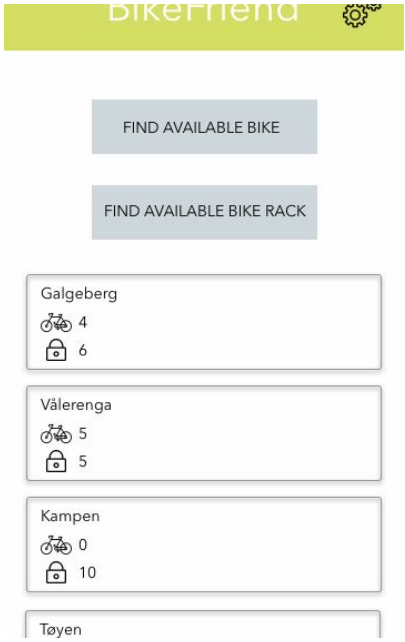
Features

- Uses a Google Map to show users available bikes and racks in their area
- Lets the user save favourites to get easy access to the most used bike stations
- Widget displaying the status of these favourite bike stations

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1

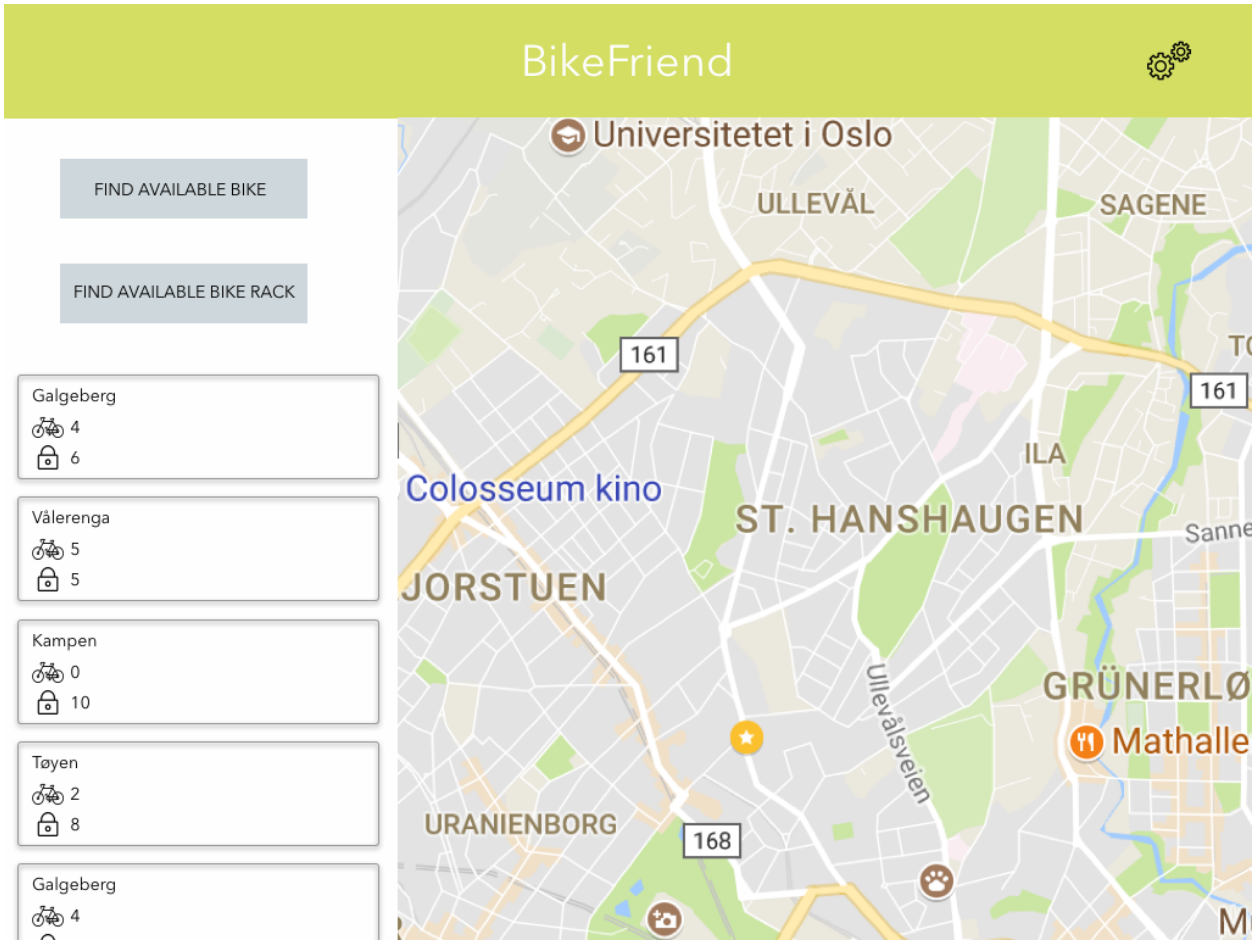


This is the main screen of the app. Here, the user can click on the buttons “find available bike” or “find available bike rack”, which launches the map activity in the appropriate mode (see screen 2). The user can also see the status of favourite bike racks in a scrollable list. If a favourite is pressed, it launches navigation to the bike station.

Screen 2

In this map, there will be markers showing bike stations fulfilling the needs of the user (i.e. if the user wants to find available bikes all markers will be at bike stations with available bikes, and available bike racks if the user wishes to park the bike). When markers are clicked, an InfoWindow with the name of the bike station, the number of available bikes and available bike racks is showed. And, when the infowindow is clicked, navigation is launched to the selected station (directions for walking when finding a bike, directions for bike when finding a bike rack).

Screen 3 (tablet)



This is a master-detail flow for tablet use, which combines Screen 1 and Screen 2. The map for available bikes will be displayed as default, and it will change to display available bike racks when pressing the “find available bike rack” button.

In addition to these screens, the user will get to a settings screen when pressing the settings icon. This screen is simply a list of favourites which can be removed by pressing an associated “red x”, and a search functions to find new favourites from the stations synced in the local database. The list of favourites is the same list as the one that will be used in the widget. There will also be a setting for how often the user wishes the status of available bikes and racks should be updated, with the default being every 1 minute (when the application is being actively used).

Key Considerations

How will your app handle data persistence?

Data in this app will come from the Oslo Bysykkel API(https://developer.oslobysykkel.no/api), which is free to use. As winter is approaching and Norway is a cold country, the bikes might be taken in for storage for winter before this app is ready, so if the api returns a status telling that the entire system is closed, the user will see a dialog box asking if he or she wants to use mock data (using local files created at a certain date before the system was closed for winter). This is of course only for testing purposes and would not be included if this app was to be released to the public. The decision to use dummy data or not will be stored in SharedPreferences, and checked before each network call is executed.

The app will use a content provider, which i find is a good fit for this project. The app will store base data and update it once every 24 hours using a job scheduler, and the number of available bikes and racks will be synced on startup and at approximately the interval chosen by the user in the settings. Sync interval and time of last base sync and data sync will be stored in SharedPreferences.

Describe any edge or corner cases in the UX.

As far as I know, there are none. This is a very simple app.

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

- com.fasterxml.jackson: to easily parse Json
- net.danlew.android.joda: for better Date use than vanilla Java
- com.jakewharton:butterknife: to easily find UI components in Java code

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

Maps: will be used to display a map to users. See earlier explanation.

Location: will be used to center the map at the user's location when launching the map activity.

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

- Create project
- Configure libraries described earlier
- Create package structure

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for MapActivity
- Create custom InfoWindow layout
- Create UI for SettingsActivity

Task 3: Create Activities

- Create MainActivity and make sure all components are ready for use in Java code
- Create MapActivity and implement the OnMapReadyCallback, OnRequestPermissionsResultCallback and OnInfoWindowClickListener.

Task 3: Implementing Google Map

- Perform the steps found here: https://developers.google.com/maps/documentation/android-api/signup to get an API key
- Insert the required components and properties in the AndroidManifest
- Implement logic handling whether or not the user has allowed location to be used, as well as asking to use this permission

Task 4: Create content provider

- Implement database and content provider the way I have previously learned in the nanodegree
- Make sure I can insert, update and query the database
- Create useful helper methods to insert, update and query

Task 5: Sync base data

- Add dummy data files to the project and create functionality to read from these
- Implement functionality to check system status from the API and select between dummy data or real data if the system is closed for the winter

- Store base data from the API in the local database
- Store status data from the API in the local database

Task 6: Show markers in map

- Add appropriate markers to map; available bikes or locks
- When pressing a marker, display infowindow with info
- When pressing the infowindow, launch Google Maps navigation to than marker

Task 6: Implement settings

- Let the user select sync interval, and store it in SharedPreferences
- Implement search functionality to let the user find and add favourites from the stations stored in the local database
- Display a list of the selected favourites, and let the user remove them

Task 7: Show favourites

- On the main screen, display favourites selected in the settings.
- When clicked, launch navigation to the bike station selected

Task 8: Implement an appropriate sync schedule

- Sync the availability status data at the interval selected by the user
- Sync base data every 24 hours
- Do the two subtasks above using a JobScheduler.

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**"