CS443 Project Documentation: MBTA Buddy

Luke Vu

May 2021

1 Project Statement

MBTA Buddy is an application that provides users with up to date train arrival/departure predictions at stops along MBTA lines. The application allows users to navigate between different lines, e.g. Redline-Braintree, Greenline-B, with an easy horizontal swipe gesture. Users can navigate up and down the line with vertical scrolling. Clicking on a stop will list the upcoming arrival/departure times for trains and their direction of travel at that particular stop.

The purpose of this project was to provide users with a fast, intuitive, light weight application for getting train scheduling. Google Maps does similar and the Android app store hosts a number of MBTA tracking type applications but all are bloated with features and take a significant number of steps to get the information you need.

This app is not intended to replace the use of those applications. MBTA Buddy is not a route planner; Google Maps would be a better alternative for planning out a trip. MBTA Buddy does fulfill a role in getting quick arrival/departure predictions.

2 Application Design

This application is targeted towards mobile phone devices but will work on other devices. It requires network services as the app makes HTTP requests for prediction data from the MBTA API.

2.1 Overview

MBTA Buddy features two activities: MainActivity and StationInfo. MainActivity displays at the top a tab layout of the different MBTA lines and a list of stops on that particular line. Horizontal swipes allow for navigation in between these different views. Vertical swipes allow navigation between stops.

On tapping a particular stop, StationInfo is started and displays a list of predicted arrival/departure times for trains and their direction of travel. An intent with the stopID and stopName is passed to this StationInfo activity. To navigate back into the MainActivity and view of the lines/stops, the user can either press the back button on their device or the back arrow at the top of the application.

Lists of the station names and station ids are stored in the strings.xml file of resources in the package structure. Each line has a list of station names and station ids. Drawables contains a mix of .svg vectors and .png bitmap iamges to display the graphics for the lines and stops.

MainActivity creates an instance of a ViewPageAdapter and adds each line's fragment to itself. Each fragment pulls from the string resources and creates a mapping of the station name to id as well as list of each stop's image id to attach. StationAdapter, extended from the RecyclerView Adapter populates the fragment with a RecyclerView of the list of stations.

StationViewHolder of StationAdapter contains an onClickListener; when a station is clicked, an intent is created with the stationName and stationID added and starts the StationInfo activity.

StationInfo first builds a mapping of line ids and their respective image source. A series of functions are called from each other starting with fetchRoutes(). fetchRoutes() makes an aynchronous HTTP GET request to the MBTA API to build a list of routes offered at the specified station. GSON is used to parse the JSON data into a data object. festingPredictions() is called and makes a request for prediction data at the specified stop. Here, buildRoutesMap() is called on the routeData object to build a mapping of the route id to the routes direction destination. This is passed along with the predictionsData to buildPredictionList() which creates the final list of StopPrediction objects that contain the direction, destination, estimated time of arrival, and image source for the stop prediction. the recyclerView is updated with this list and the view is created.

2.2 Technologies Used

- ViewPager - This is layout manager that allows users to flip left and right through pages of data. Here I use it to navigate between different fragments containing the layout of the MBTA lines.
- Fragment - Each MBTA line has a fragment class that provides the data to populate its view
- RecyclerView - This view allows for dynamically displaying the stops on each MBTA line, every fragment contains a recyclerView.
- OkHTTPClient - This library is used to make asynchronous HTTP GET requests to the MBTA API to retrieve JSON formatted prediction and route data
- GSON - This library is used to parse the JSON formatted data into data objects to retrieve the target data.

2.3 Layout Architecture

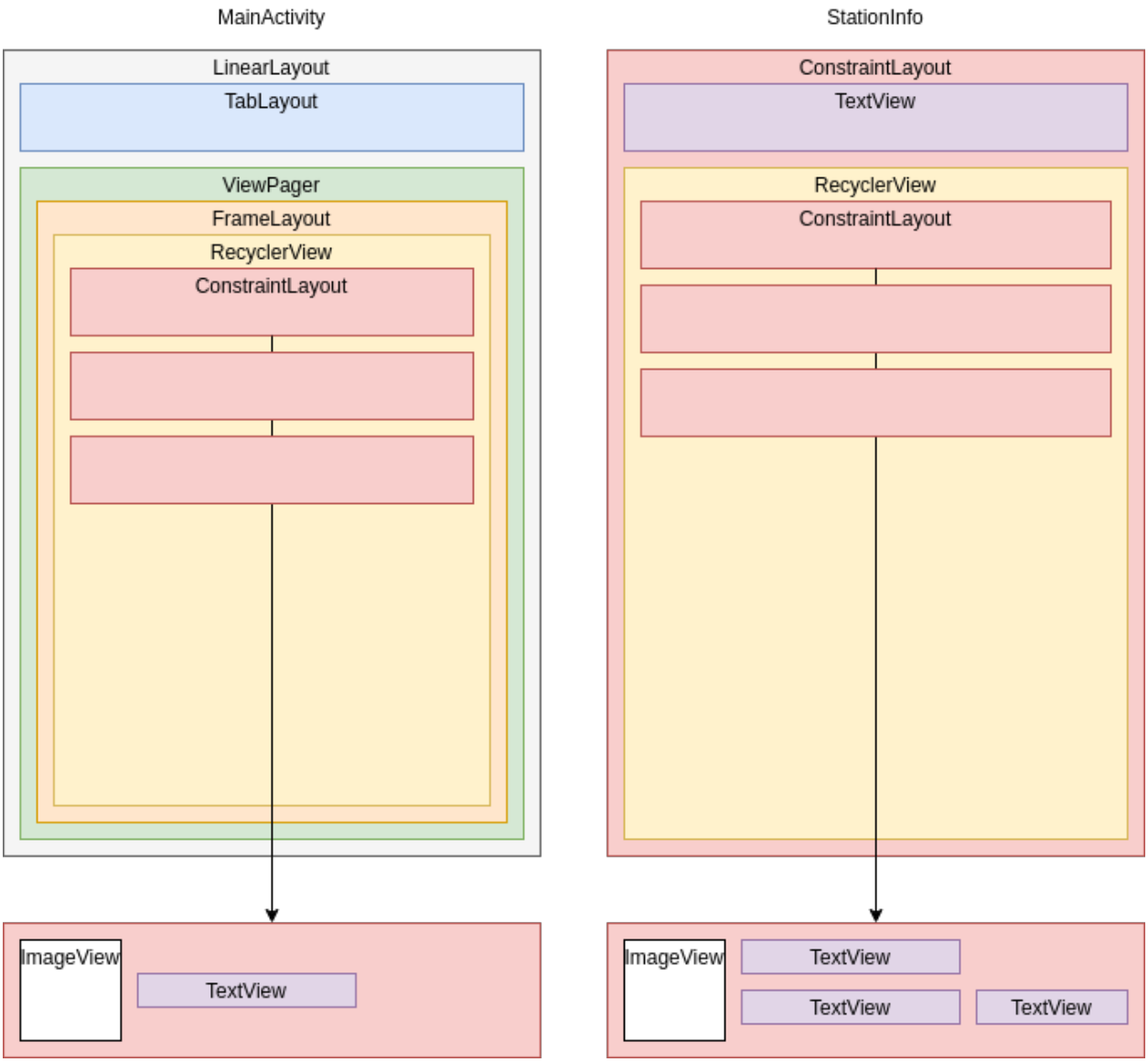

Figure 1: Layout Architecture

2.4 Class Diagram

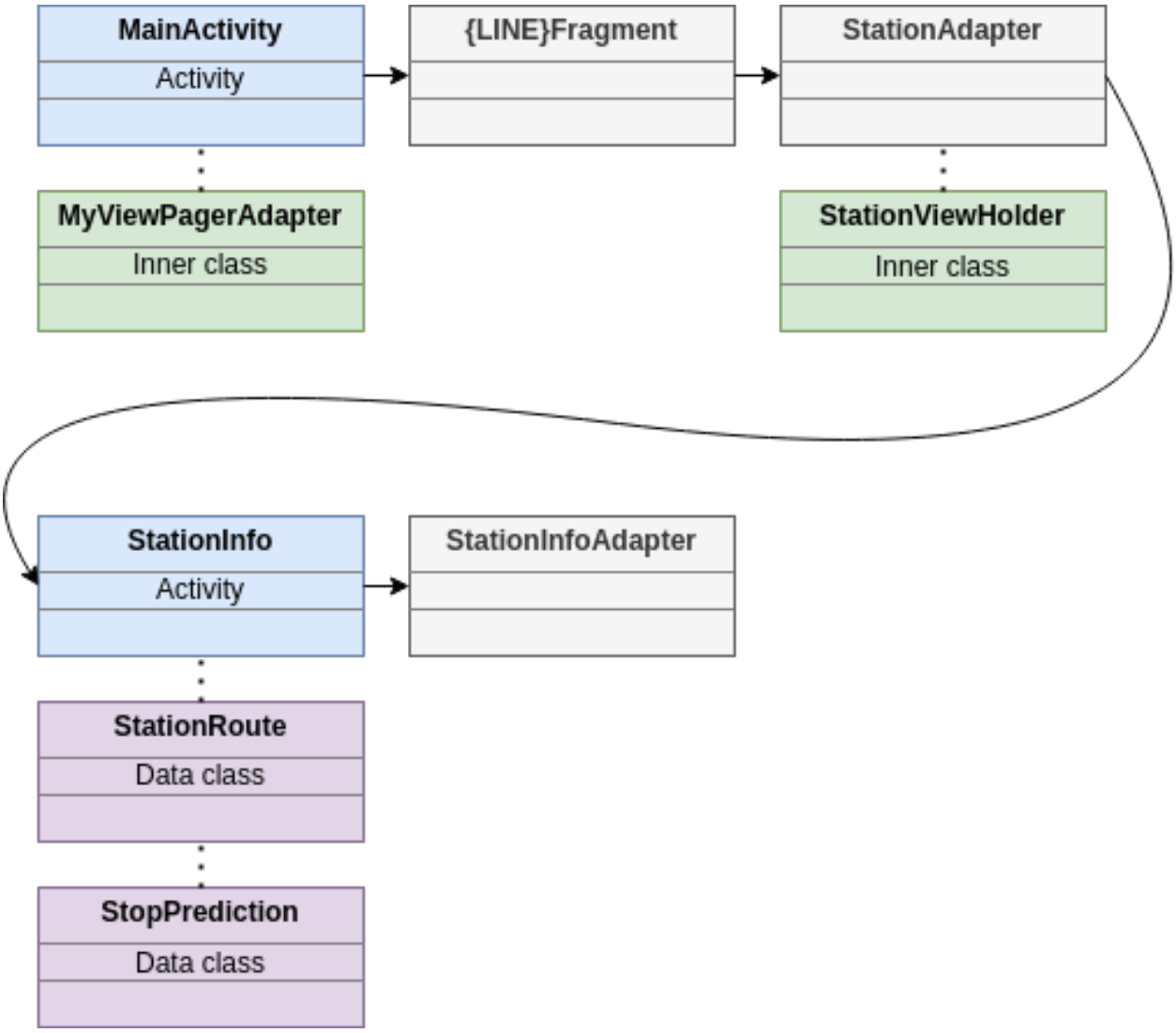

Figure 2: Class Diagram

2.5 StationInfo Program Flow

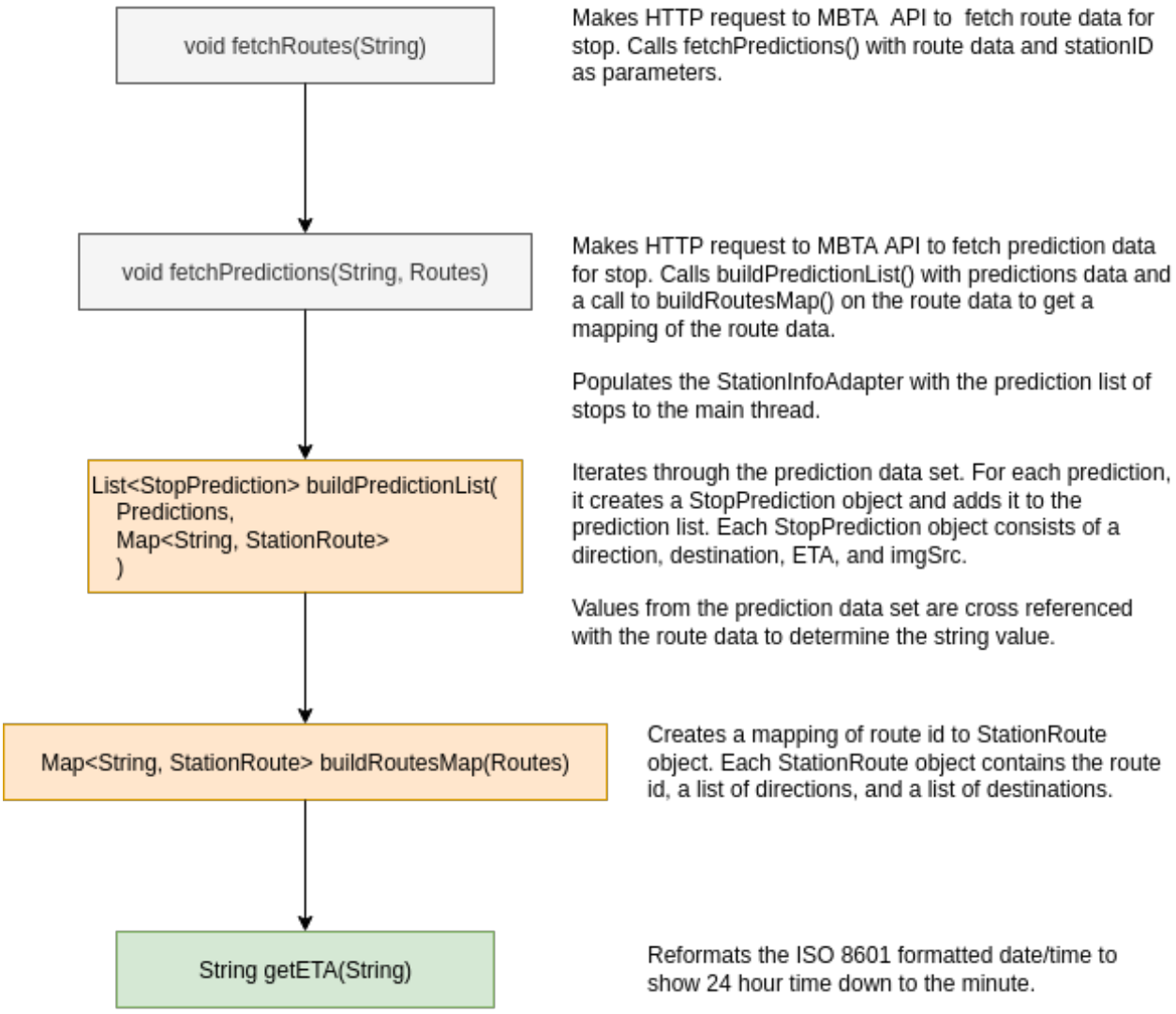

Figure 3: StationInfo Flow

3 Application Implementation and Evaluation

The application was built on and tested for Android API 28. Prediction data will be available as long as the app is used during times when trains are running. During initial testing, the app was crashing on certain stops, namely stations where routes started and ended. This was due to nulls in the arrival field of the prediction data. Trains that originate from these stations do not have a arrival time because they start there. I fixed this by adding a null check and choosing to use the departure time as an ETA instead. Some stations come up with no predictions; this is due to the station not currently being serviced by a train line for construction or some reason. The application does not crash here but simply delivers an empty list of predictions.

3.1 Use Case

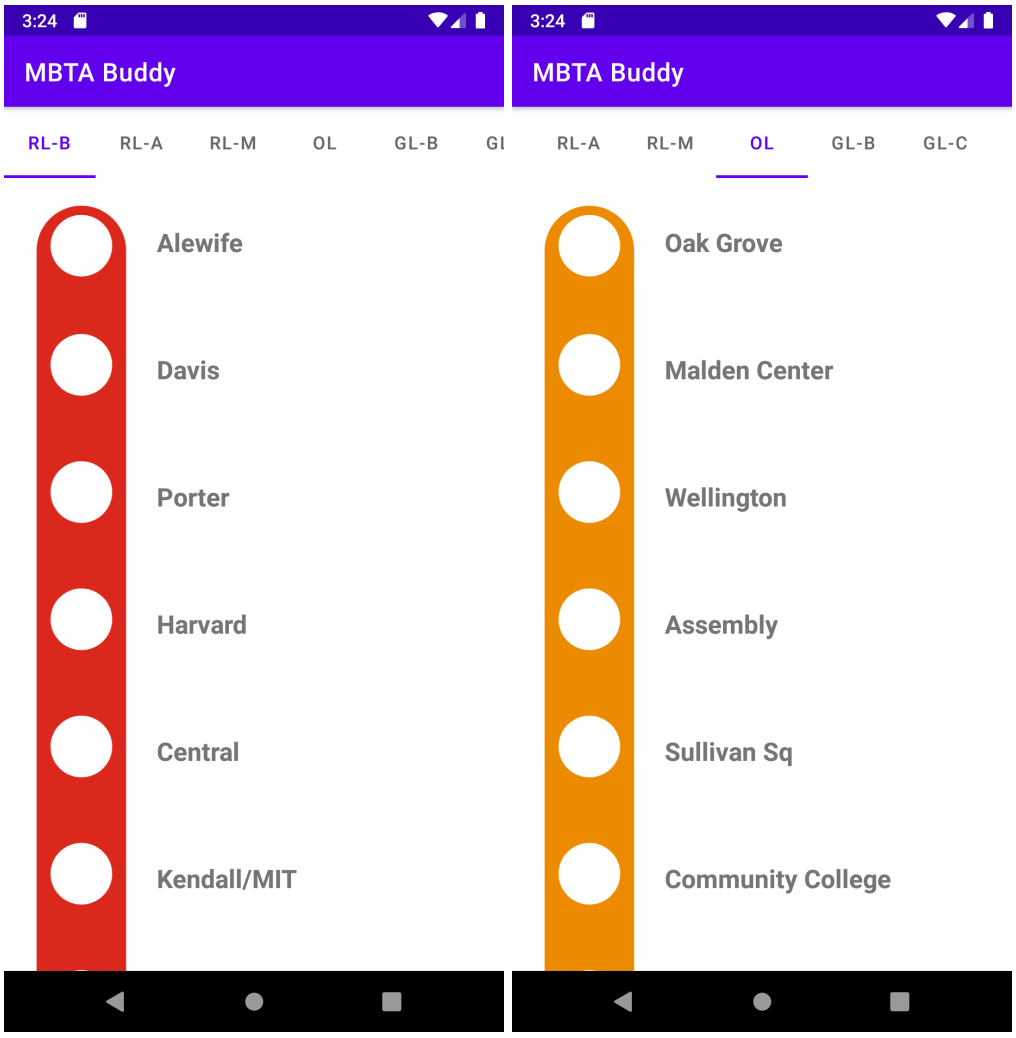

Figure 4: Users can swipe left and right to change lines

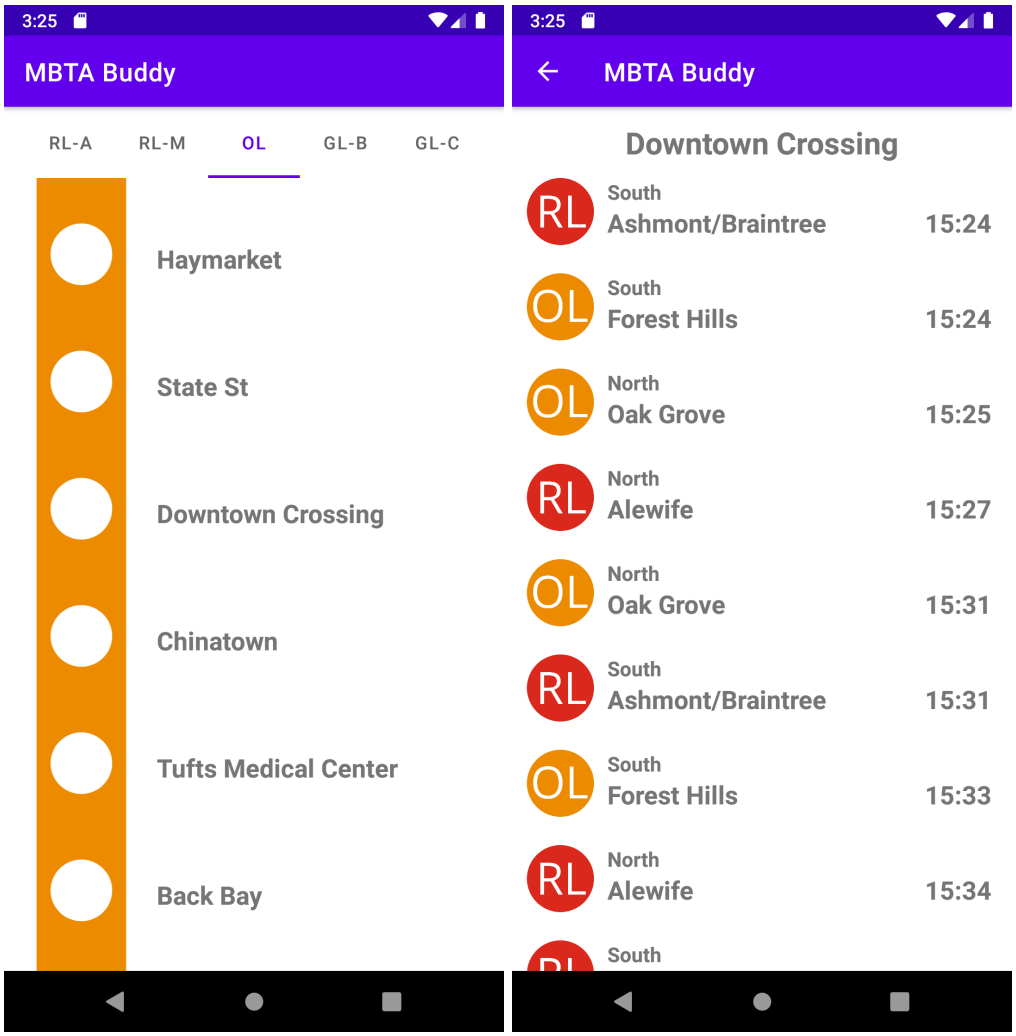

Figure 5: Users can swipe up and down to navigate the line — StationInfo Activity view

4 Experiences Thoughts

The application was built on a time crunch without a clear plan for how I wanted to implement the specified features. I ended up using ViewPager for navigating between lines as I thought this was the fastest, most intuitive way to navigate the fragments. I decided to use OkHTTPClient as a library to implement my API requests instead of Async tasks because that method has been deprecated. For the most part, I tried to use current libraries and functions and whatever functionalities I could get up and running.

With more time, more features could be added. The MBTA API is rich with information that can be delivered, including outage information, accessibility, and real time tracking of vehicles. The current application only tracks trains (subway/trolley) but support for buses and other means of transportation can easily be added.

Originally, I had planned for some location services to automatically pull up station info for the closest station on application start but was pressed for time and dropped this feature.

I enjoyed this class as it provided an opportunity to do some different programming then what I was used to and build an application from the ground up. I would have liked to see deprecated tools/libraries/functions not be taught, and instead whatever their replacements were be used. Kotlin was fairly easy to pick up coming from a strong Java background and was enjoyable to work with in building out this project and completing the homework assignments.

I would have liked to have more homework assignments, I learned a lot from doing each one, debugging the programs with Android Studio.