Discovr - Design Document

By Simon Jinaphant, Yueyue Zhang, Valerian Ratu, David Wong, Nicholas Leung, Jacqueline Choi

INTRODUCTION

Discovr is a mobile app for UBC students and its residents which allows them to discover and navigate to events and happenings on campus. It also provides them with a graphically interactive map containing relevant information that cannot be easily found or accessed from a search.

This document iterates on our choice of implementation and the architecture of our program.

SYSTEM ARCHITECTURE AND RATIONALE

Main Components:

1. A mobile platform for users to interact with and display information
2. A mapping service which provides map data for visualization of location and direction
3. A web-crawler that obtains event data from specific websites
4. A database service which can be accessed for information about events and buildings
5. A service to provide information about public transport schedule and private timetable

Android

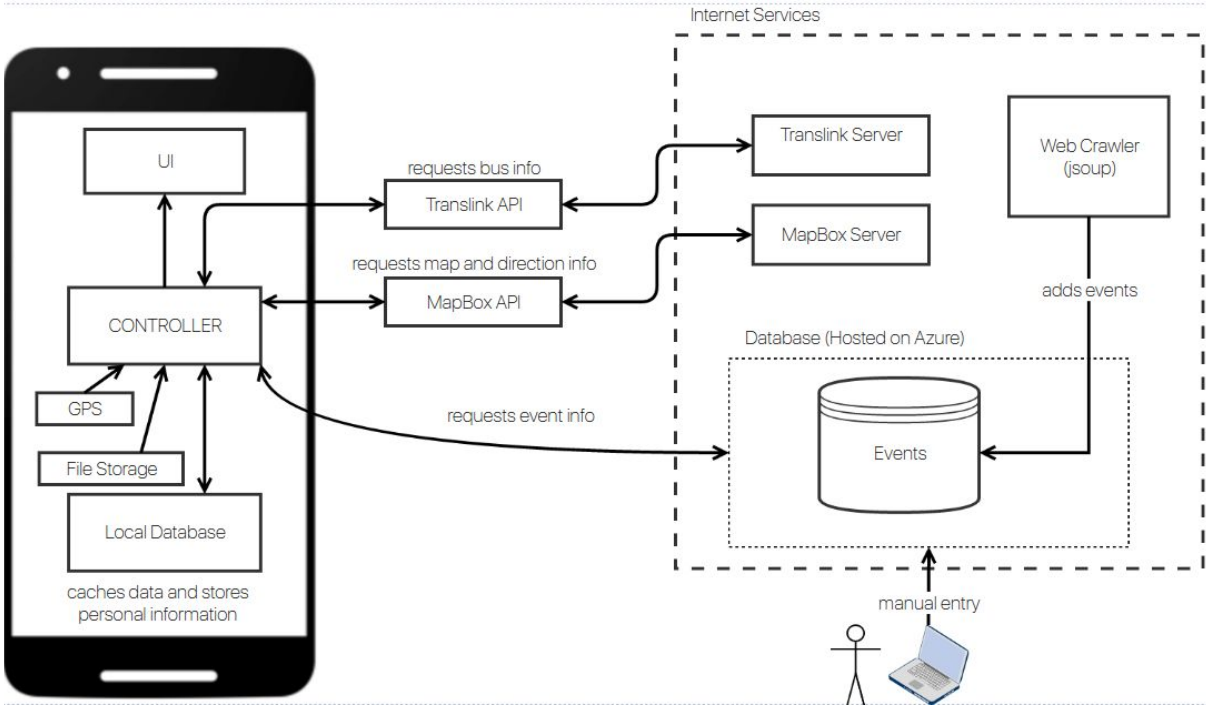
Android is a mobile operating system owned by Google. Android is used on many smartphones and portable devices which makes it a good target due to the location-centric nature of our application. Development in Android is done in Java which allows us to access powerful and tested libraries and APIs to implement the app's functionalities.

Mapbox

Mapbox is a mapping framework for developers. Mapbox provides mapping functionalities such as visualization and directions. It allows for custom styling and layering to personalize maps to a specific application. Mapbox is also a tool for creating datasets in GeoJSON format.

Mapbox Android Services allows our application to integrate with the Mapbox API as well as the Directions API. This provides us with the necessary methods to display custom maps and routing information from one point to another.

Dynamic View



Translink API
Translink provides us with real-time information regarding bus and skytrain routes to assist in providing directions to and from UBC. The Translink API includes methods for us to search for nearby stops based on location, next bus estimates for a given stop, and the real-time status of a particular bus.

jsoup
Jsoup is a library that allows us to extract data from the HTML structure of a webpage. We use this to obtain events from ubc events page and submit it to the remote events database.

Android Asynchronous Http Client
Android Asynchronous Http Client also known as loopj allows us to perform sychronous or asychronous http REST calls. We use this to gather events from our remote database and submit REST calls to the transit API.

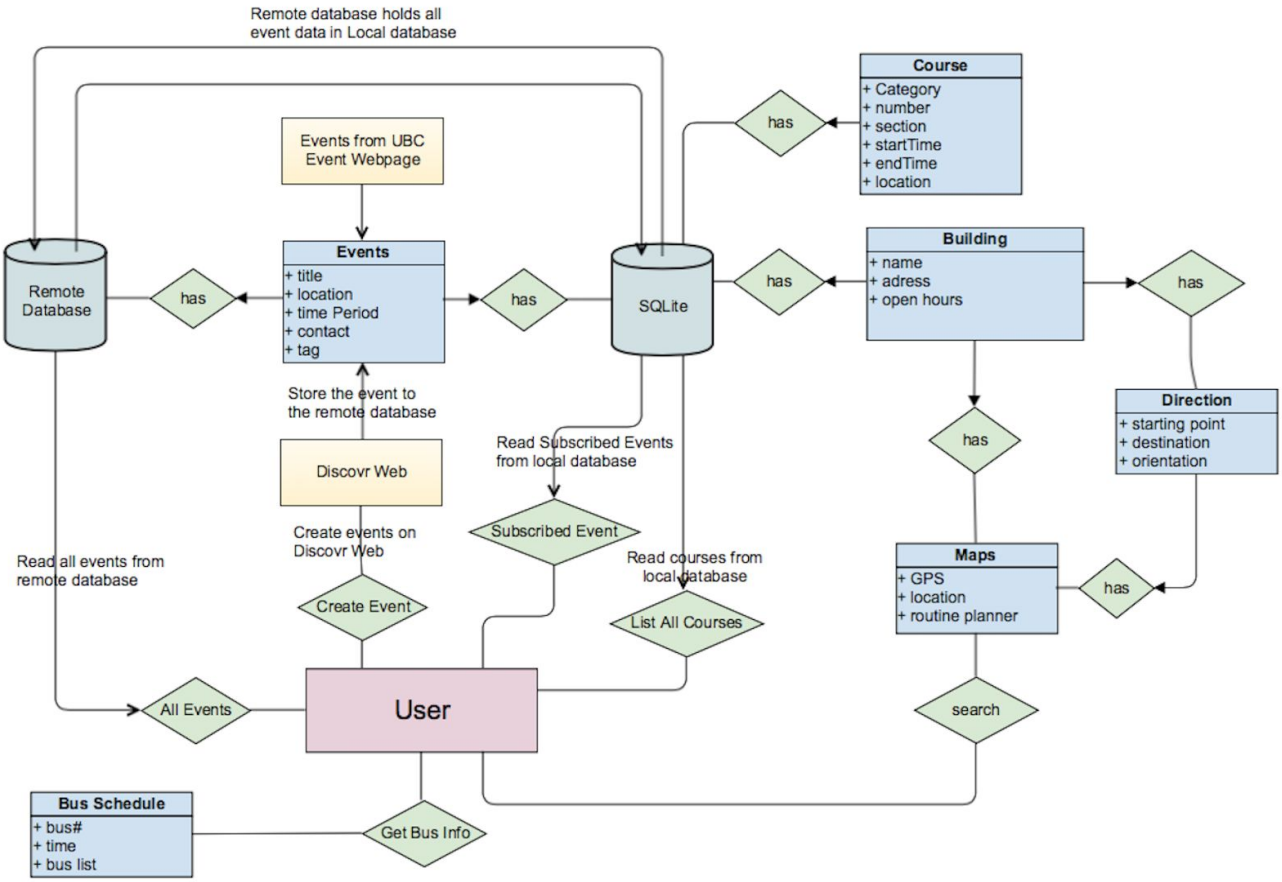
Microsoft Azure
Azure offers a flexible cloud platform to deploy and manage SQL databases, servers, and web services from. Azure will allow us to store and retrieve information that need to be frequently updated and uniformly distributed to all users; these information include campus events and construction data. We primarily chosen Azure over other cloud providers due to previous experience and familiarity with the platform.

ICal4j API
ICal4j provides us with methods to read data from local ical files and parse the data to our program. The iCal4j helps us store the parsed information as "Course" objects in our program and display them on our application.

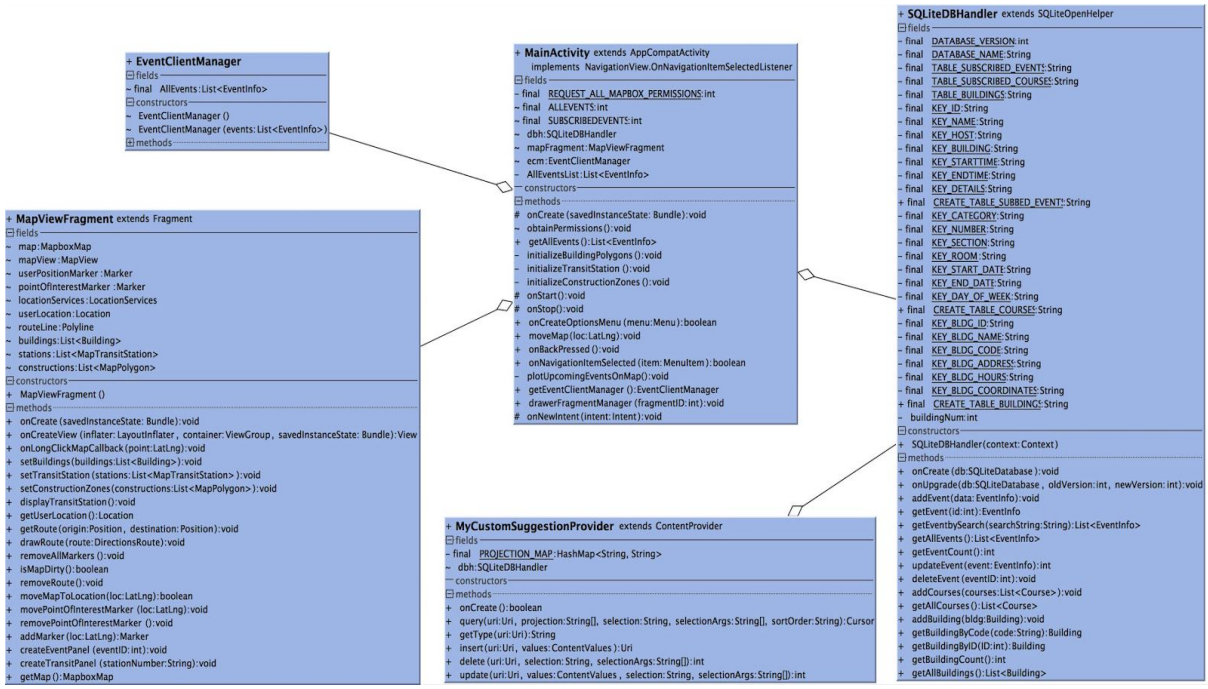
Robolectric
Robolectric is an Android testing framework which allows us to perform tests on our local machine instead of on actual Android system/emulator. This helps us perform integration testing without having the need to mock the Android environment.

Data

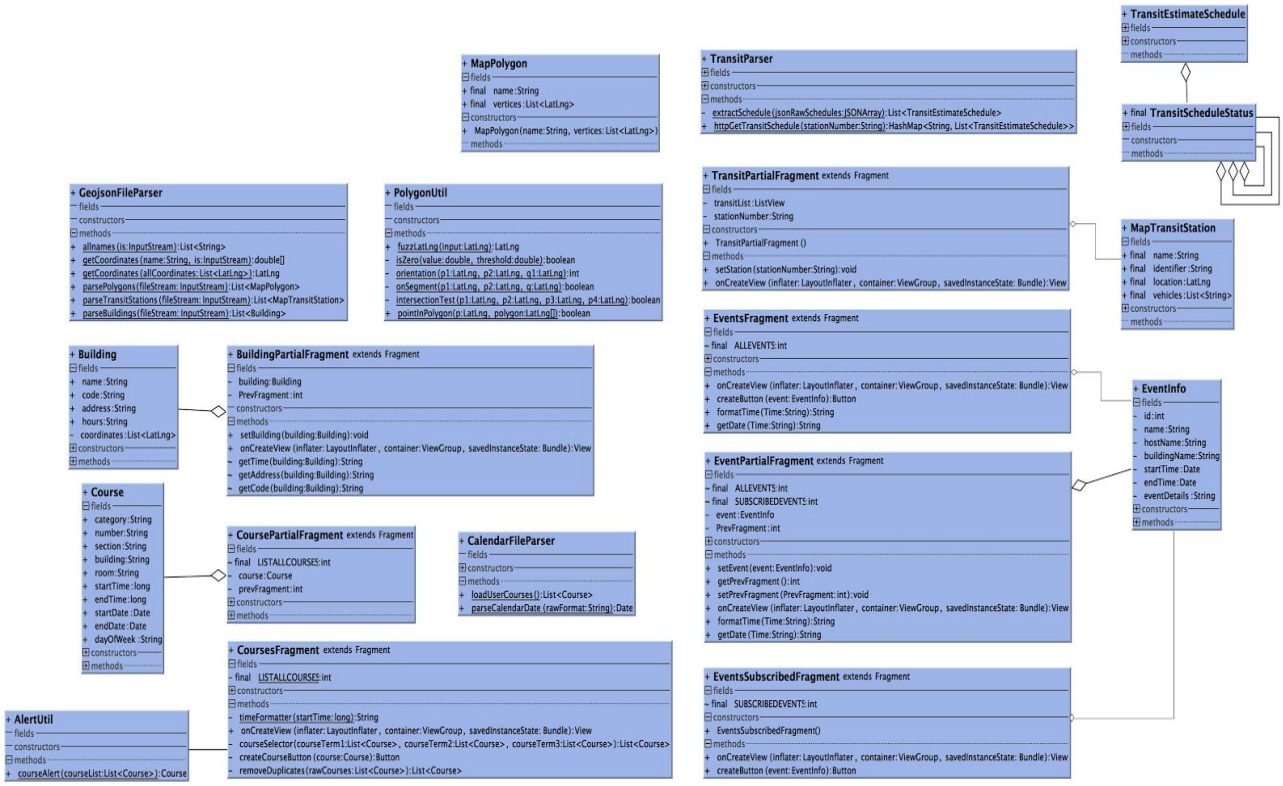
The data flow of our application is shown below:



Class Diagram

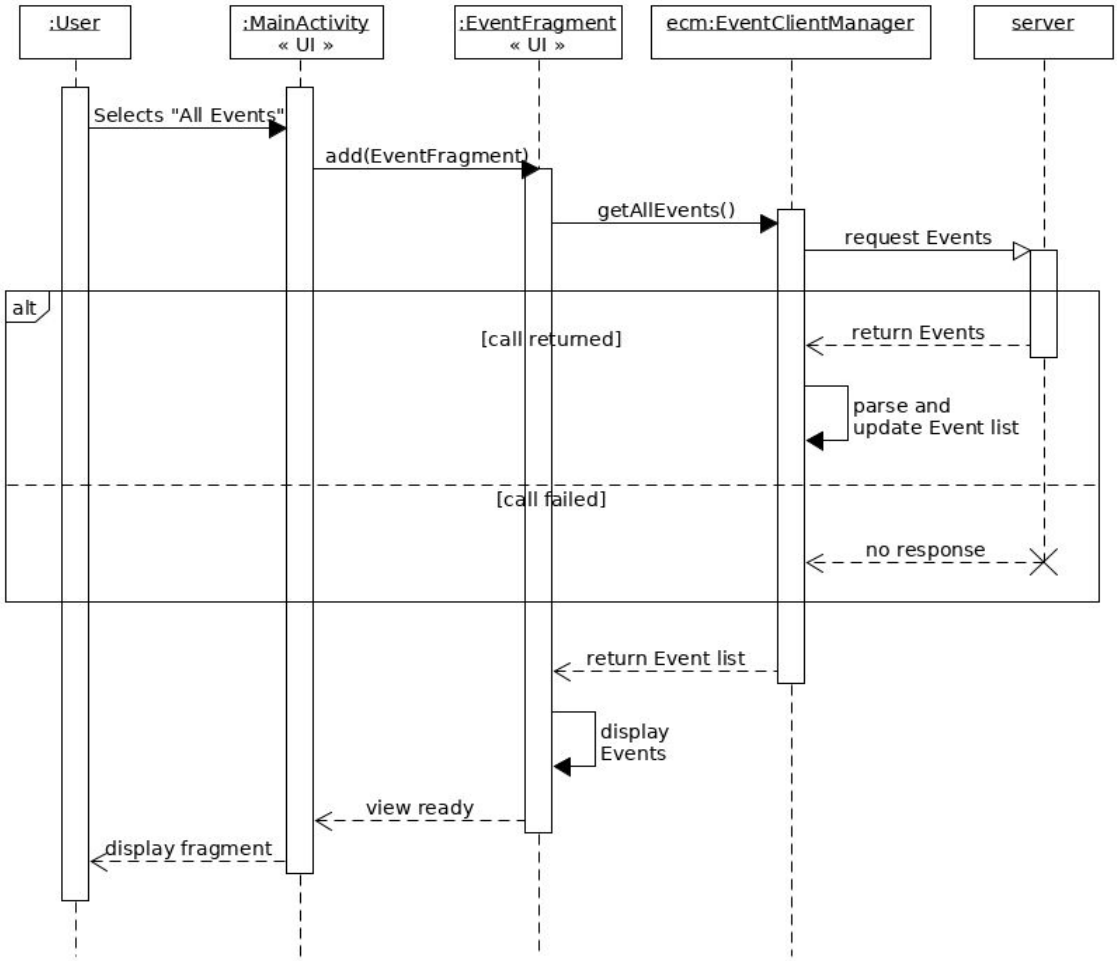


MainActivity is the top-level activity class for Discovr. It associates with SQLiteHandler to handle information stored in local database and interacts with EventClientManager to manage events from the remote database. One of our main features is to provide users with a graphically interactive map, so we set the map view as the background for our GUI. The association between the MainActivity and MapViewFragment helps us accomplish this. MainActivity also holds the interaction between low-level activities and user' action, and specific activities are implemented apart from the MainActivity, which is shown in the diagram below.



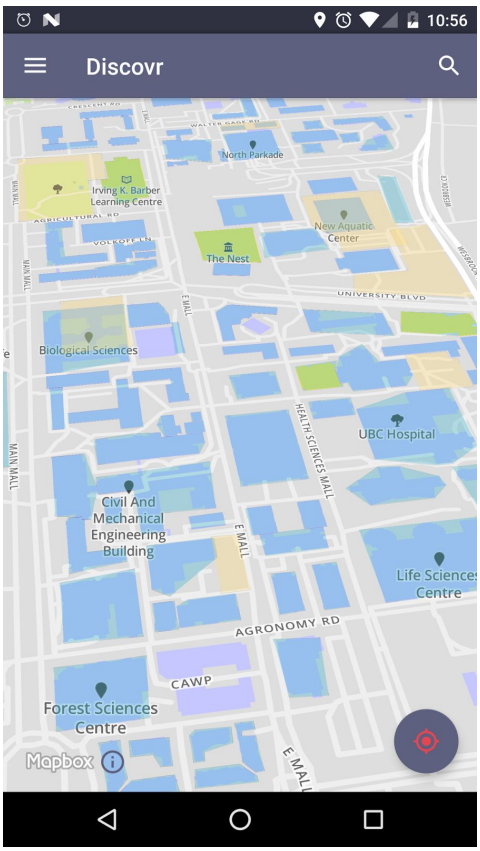
The main models implemented in our project are EventInfo, MapTransitStation, Building, and Course. The GeojsonFileParser parses building information from json files and stores the parsed data to the local database as a "Building"; CalendarFileParser parses local ical files and stores the parsed data to local database as a "Course". Each model shown above is associated with its corresponding fragment or partial fragment. The CourseFragment, EventsFragment, EventSubscribedFragment are matched with the buttons on the navigation drawer.

Event Server Interaction



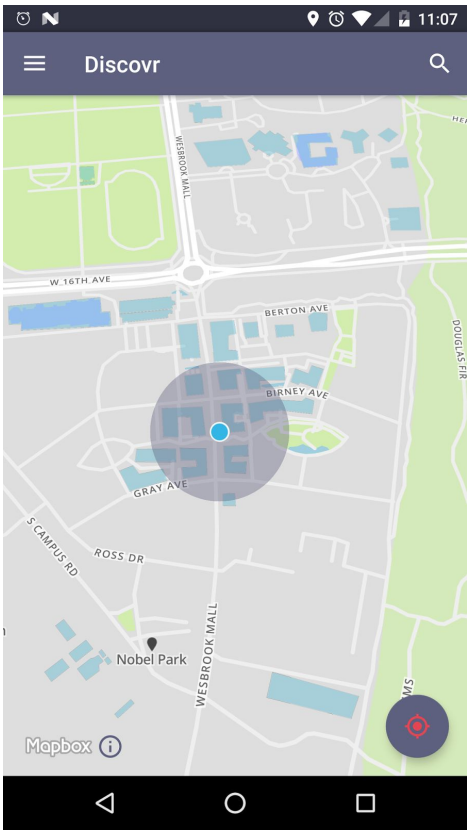
This sequence diagram shows what occurs within the app when the user selects All Events from the navigation drawer. This call is passed through MainActivity and passes it to the EventFragment class which is responsible for displaying information about the events. This fragment then calls an instance of an EventClientManager which is in charge of sending http requests to the server and keeping an updated list of events. If a list of events is returned from the server, the EventClientManager updates its internal list of events and returns that to the caller fragment to be displayed. If there is no response from the server, EventClientManager sends the most recent list of events it recieved previously.

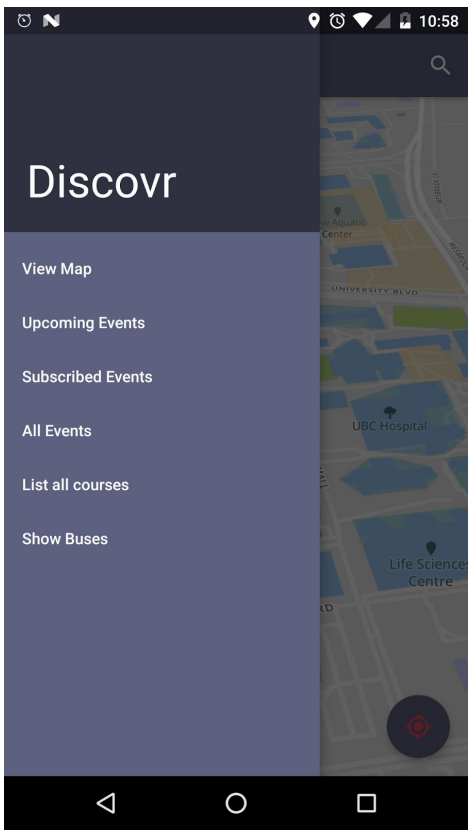
GUI



The landing page of our application is the view of a map which depicts the UBC campus. The map is fully interactive, with various elements being clickable.

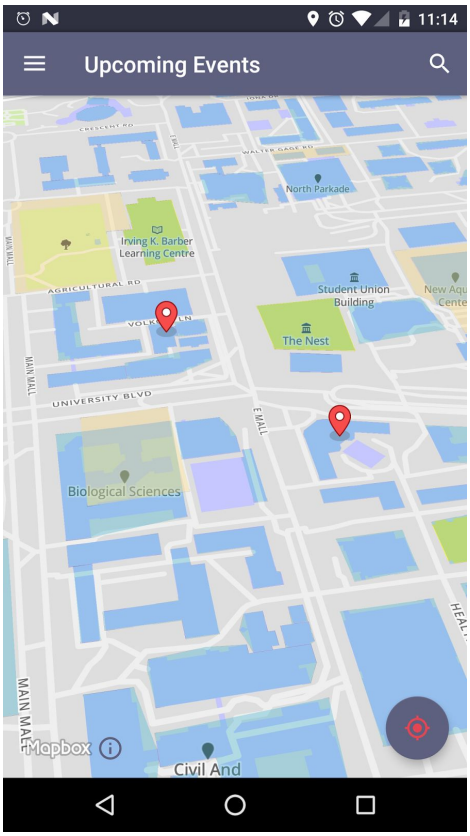
For example, if a building is selected, a pop-up appears with information about the building.

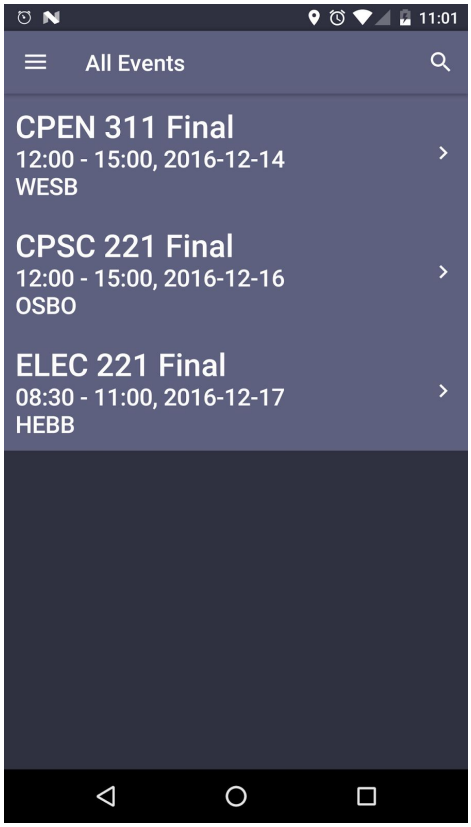




Pressing the icon on the bottom right centers the camera on the user's current location if it is available.

The features of the app can be reached through the navigation drawer which appears by either pressing the hamburger icon on the top left or by swiping across the screen from the leftmost edge.

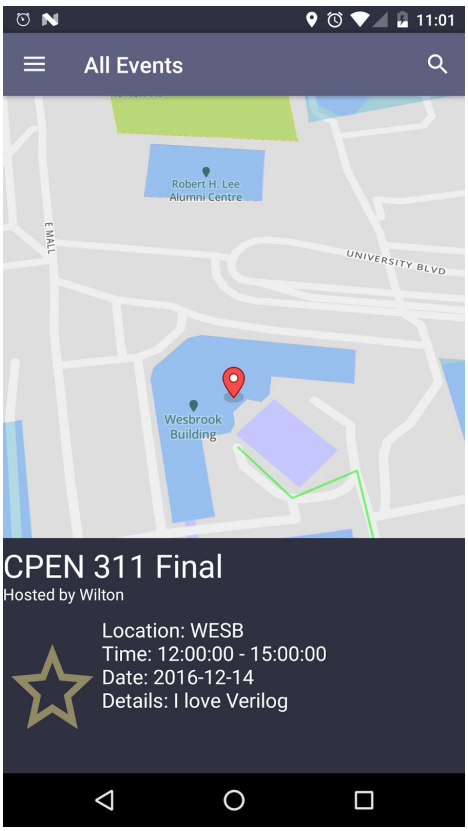


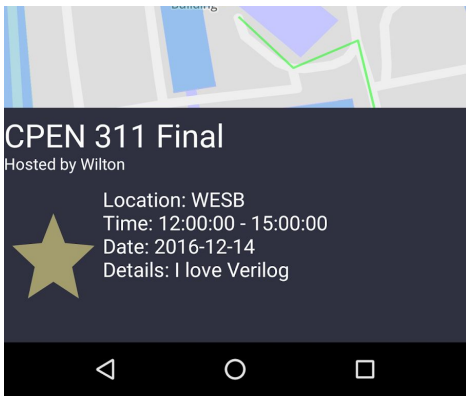


Accessing the list of Events can be done through the "Upcoming Events" panel or the "All Events" panel.

The Upcoming Events panel only displays events that have a location and are occuring within 24 hours from the current time. It is shown as markers on the map which can be clicked for more details of the event.

The All Events panel displays all the events that haven't ended. This panel is shown as a list with a preview of the event details.

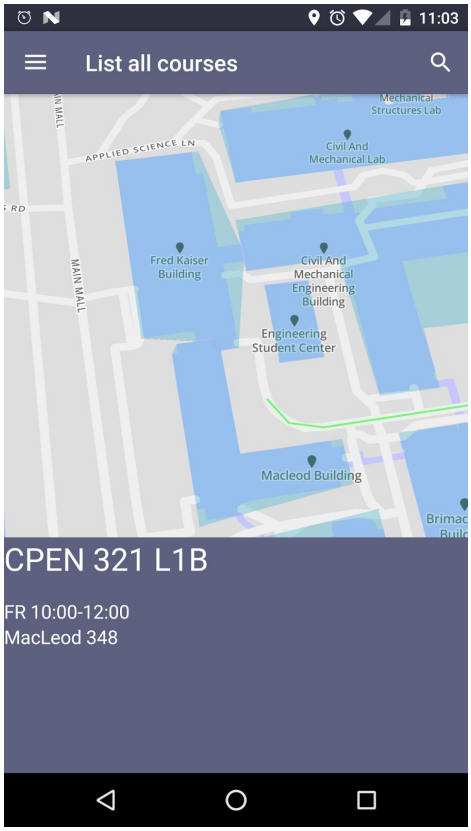




By selecting a specific event, an informational pop-up is brought up and displays detailed information about the event.

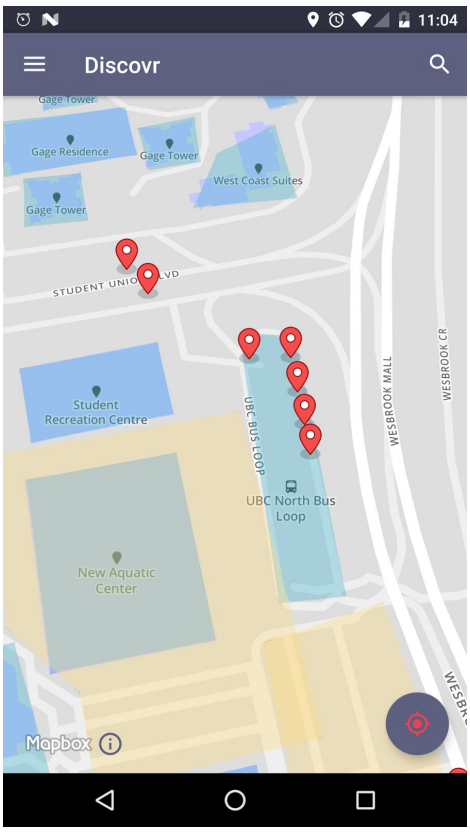
To subscribe to an event, the user selects and fills the star icon.

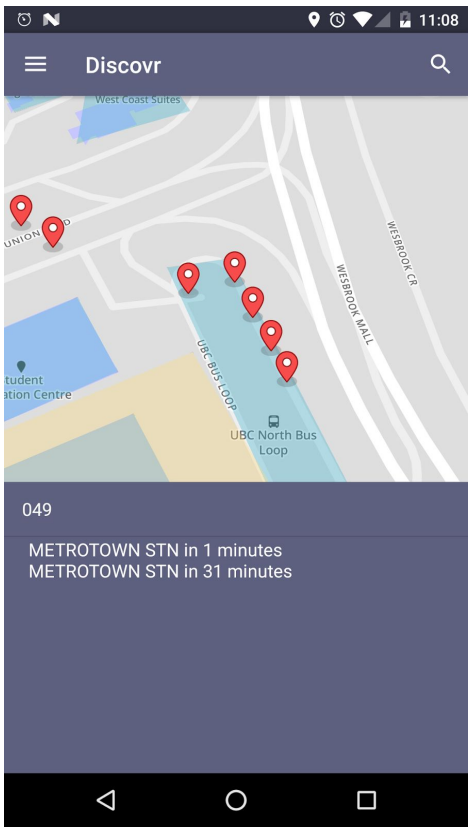
Once the star is filled, the event will appear under the Subscribed Event panel and can be accessed more readily by the user.



The List All Courses panel appears when the user has a valid course ical file within their Downloads folder. This file is parsed and the courses the user is taking is shown on this panel.

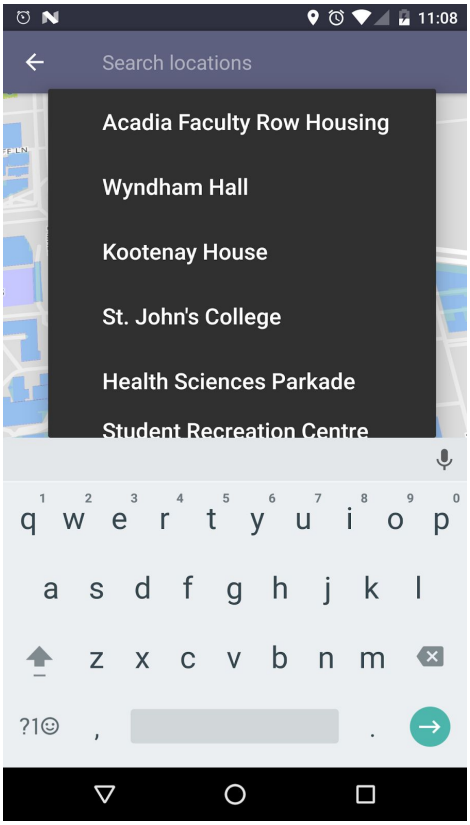
To user can select a specific course to get more information about it. This brings the map to the location of that course alongside an informational panel with the course specifics.

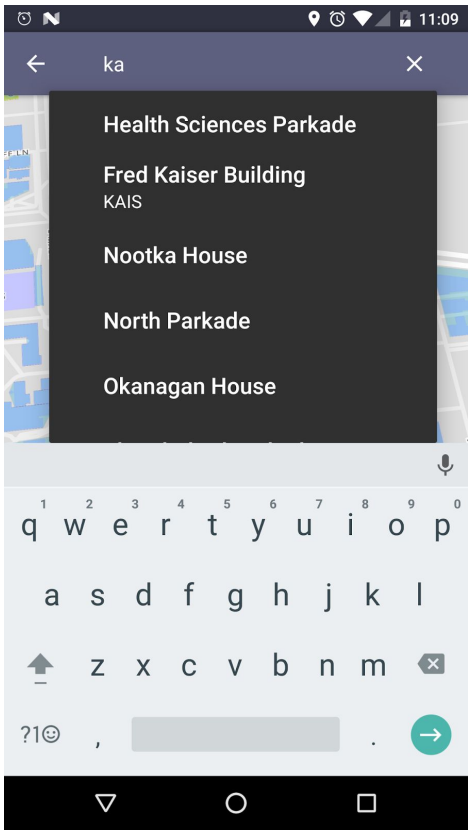




The Show Buses option brings the user back to the map view with markers on the map where bus stops on campus are.

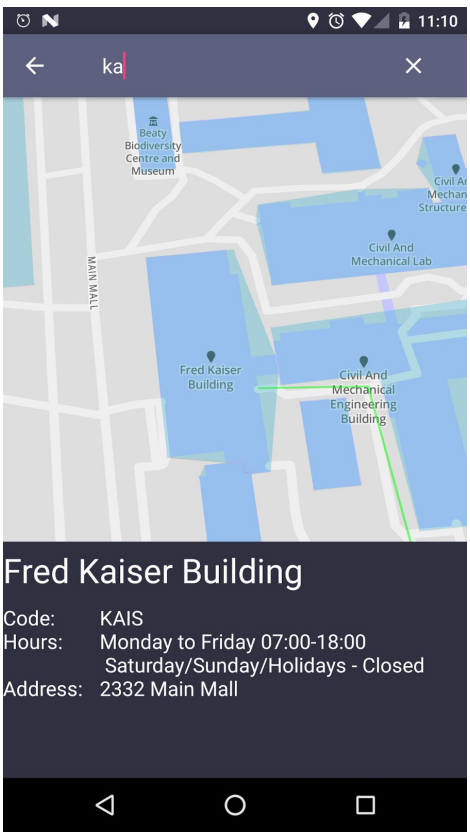
By selecting a marker, the user can get information about that bus stop such as which bus stops there and at what time the bus will arrive at that stop.

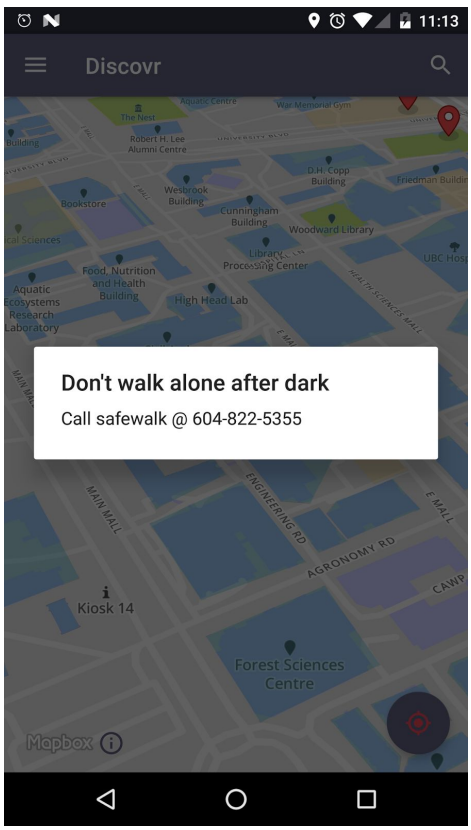




User can also select the search icon on the top right to get to the search interface.

The user is able to search against all the buildings on campus and autocomplete suggestions are provided to the user as they search.





Upon a successful search, the map is centered to the building of interest and an information panel is also provided.

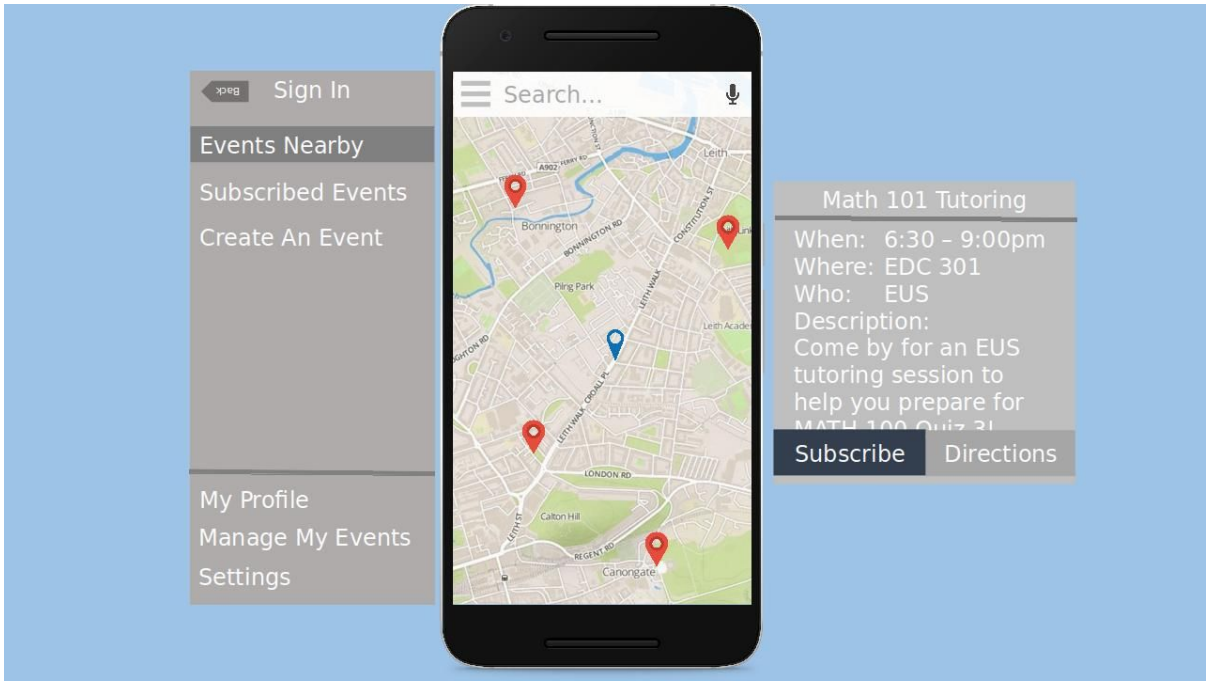
The application can also provide context aware notifications to the user. If the time is after 9:00pm the user is prompted with a safewalk notification.

The user also recieves notification if they have a class that starts within ten minutes.

Validation

The UI is focused on meeting both the functional and nonfunctional requirements as outlined in the requirements document. Each use case influence how the UI is structured to make the flow of program usage as natural as possible. Through feedback with potential users, we found that a simple and intuitive interface was prefered over a more complex detailed interface.

To validate the initial GUI design, we created UI mockups which approximates what the app would look like:



The mockup would have animations that walks the user through a use case to obtain feedback. Our design is heavily influenced by the standard Android patterns and we found that users were able to navigate through the app with little to no instructions.

Some of the feedback that we had include opinions on whether events should be displayed as a list or as items on the map. One thought that we should implement "Upcoming Events" rather than an "Events Nearby" functionality which is more relevant to the user. Through this we decided to implement "Upcoming Events" as suggested and that this option will represent the event list as markers on the map which will then provide the user with proximity information about the events. Furthermore, we decided to keep all the other event related lists as actual scrollable lists rather than markers to keep the map interface more manageable and clean. It follows that one of our major focus was to balance the display of information to keep the design simple while still allowing users to find relevant data within a reasonable time.

We also modified our requirements and use cases to meet the constraint of the project. As features are modified or removed, we discussed whether it still reflect the needs of the user. For example, we removed the search by tag functionality when dealing with events. This makes it harder for the user to pinpoint the types of events which they like; however, they can still subscribe to events to have them on a separate list. Users mention that they would like to have a searching and filtering capability, but time constraints prevent a proper implementation of that feature and if given time we would like to extend this functionality in full.