**CS673 Software Engineering** 

**Team 1 - Hoopfinder**

**Software Design Document**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Saloni Rawat | Team lead, Config leader |  |  |
| Mike Zhong | Back up team lead, Design and implementation leader |  |  |
| Jamie Smart | QA Leader |  |  |
| Sriram Doss | Requirements leader |  |  |
| Douyao Zhang | Security leader |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **01** | **Saloni Rawat** | **3 Oct 2019** | **Initial draft** |
| **02** | **Saloni Rawat** | **11 Oct 2019** | **Updated logo, introduction** |
| **03** | **Sriram Ramdoss** | **17 Oct 2019** | **Updated Database - Firebase** |
| **04** | **Mike Zhong** | **24 Oct 2019** | **Added use case diagram and class diagram, expanded notification package explanation** |
| **05** | **Mike Zhong** | **04 Dec 2019** | **Added sequence diagrams**  **Added more packages**  **Added design patterns** |

[Introduction](#_heading=h.gjdgxs)

[Software Architecture](#_heading=h.30j0zll)

[Design Patterns](#_heading=h.1fob9te)

[Key Algorithms](#_heading=h.3znysh7)

[Classes and Methods](#_heading=h.2et92p0)

[References](#_heading=h.tyjcwt)

[Glossary](#_heading=h.3dy6vkm)

# Introduction

*In this section, give an overview of this document, and also address the design goals of your software system.*

Hoopfinder is an android application that allows users to connect to other nearby interested players to organize events at nearby courts. More details on the project are available under the [SPPP](https://docs.google.com/document/d/1xOXwoxuaSoGwdceJBj4nmJrN1tKBkyNZ/edit)

The purpose of this document is to provide documentation to be used in the design and development of Hoopfinder. SDD - Software Design Document - is a design description of the system to allow for the development to proceed with an understanding what is the main expectation[[1]](#footnote-0).

The main design goals for the projects are:

1. Identify a database to securely store the user information as well as the location and court details. After an initial research on SQLite, the team decided to move forward with Firebase to store the user credentials. The data regarding the court location will also be saved on Firebase. We will also be storing the relations between users and between user and courts ( Which user follows which other user /court)
2. Easy Integration: Since this will be the first time that the team will be working on android development and its components, it is important that the integration of individual component developed by the team member is seamlessly and doesn’t lead to additional error and efforts.

# Software Architecture

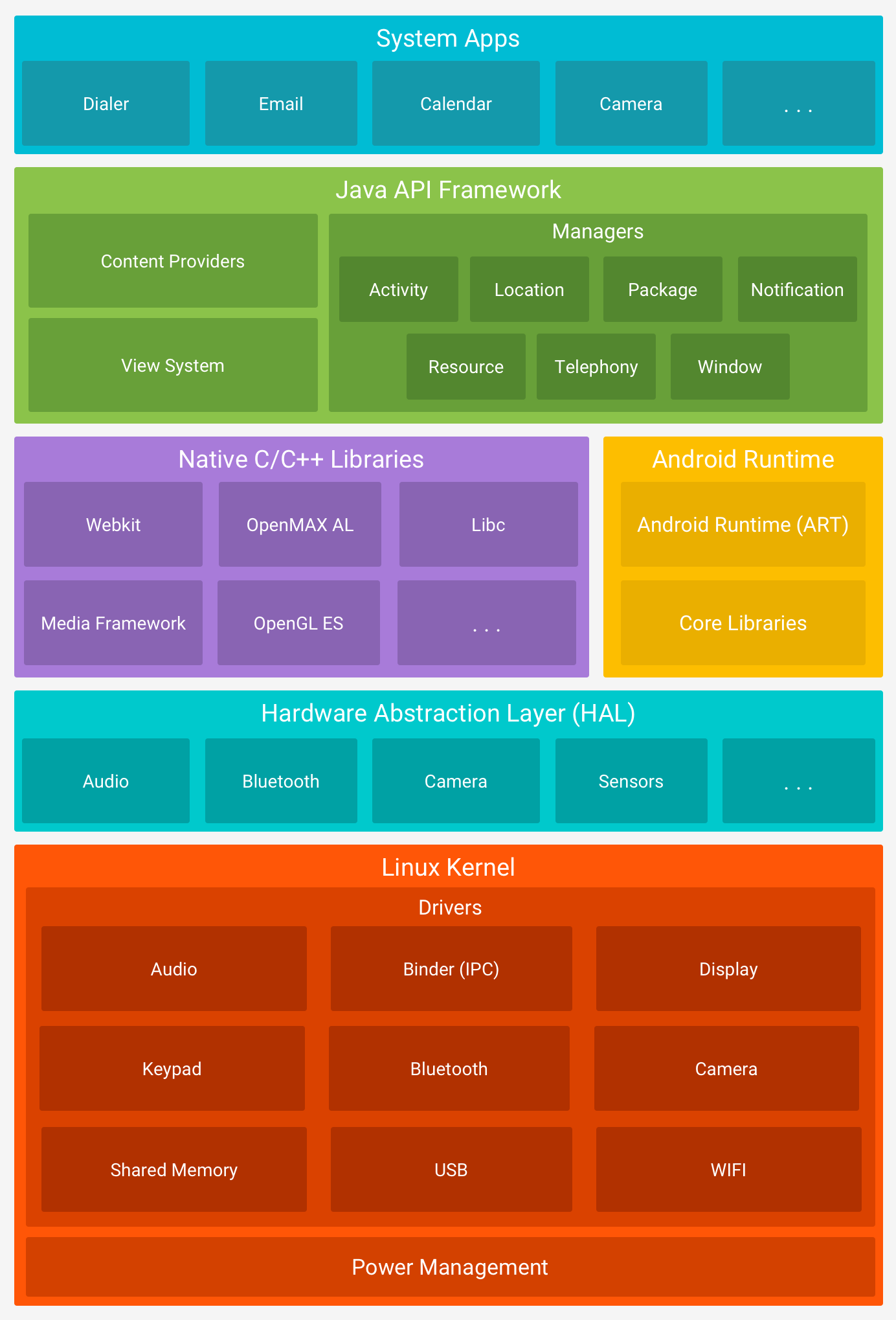
*In this section, you will describe the decomposition of your software system, which include each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagram for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used.*

As mentioned earlier, the application will be built on android platform. Android is an open source, Linux-based software stack which can be used by a number of different devices[[2]](#footnote-1).

## Architecture Diagram

## Android Platform

The following diagram from Android developer, aims to elaborate on the major components of the platform.

**Figure 1.** The Android software stack.

## The app is built using Android Studio, where a new activity is created for each user story where user interaction is required.

.

## Database - Firebase:

****

For storing personal information as location information, we will be using Firebase.

Firebase provides a realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase

Authentication - It can be handled both for username/email/password and for social authentication

**Authentication:**

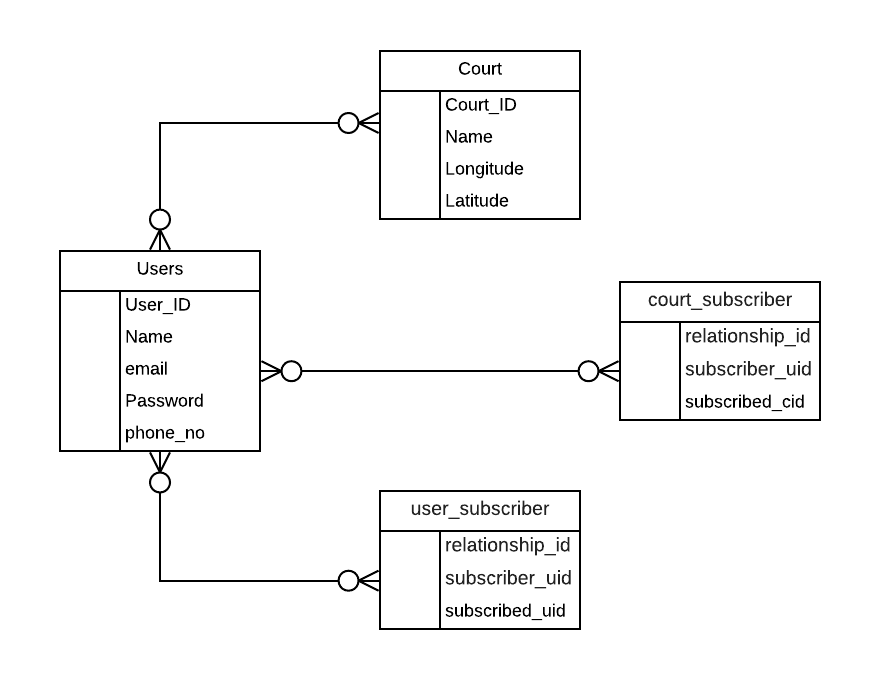
For authentication purposes we will be using Firebase authentication, which will allow users to create their own login using their email info. This will be especially useful when the user needs to search or find another user. Firebase authentication makes use of another database instead of the real-time database. To ensure consistency, important user information will be duplicated in the real time database

Firebase Security Rules -It’s stand between your data and malicious users. You can write simple or complex rules that protect your app's data to the level of granularity that your specific app requires.

Google Analytics - It is a free app measurement solution that provides insight on app usage and user engagement.

Realtime Database - Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime.

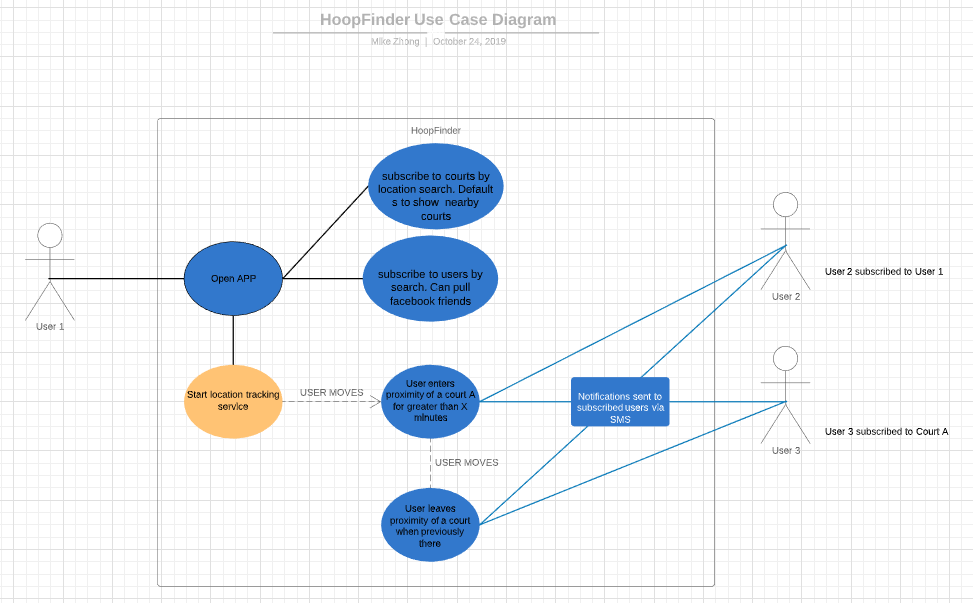
Apart from authentication, we will be using the firebase for storing the following information as well.



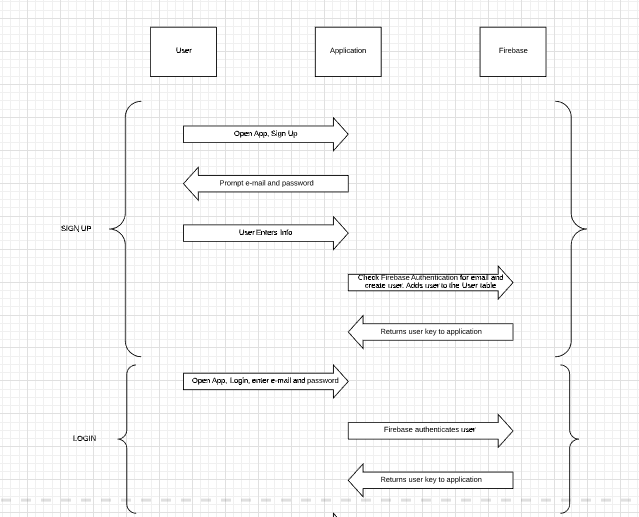
## Packages

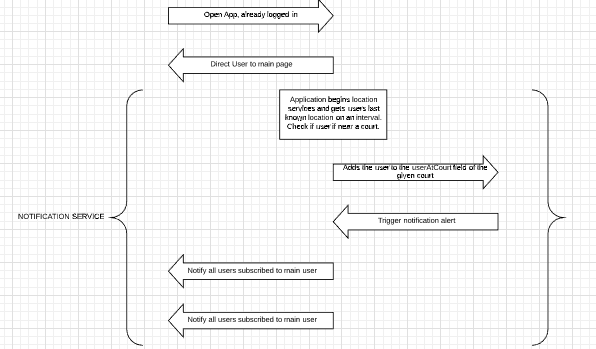
|  |  |  |
| --- | --- | --- |
| **Name** | **Purpose** | **Comments** |
| App | Contains   * Facebook login * User location information. * User information |  |
| Notifications | Classes managing notifications, frequency of notification, notification when user leaves. Class will also control how long a user must be at a court before a notification is sent to prevent “false positives” (e.g- user drives by court and a notification is sent out) |  |
| User | Data model for the User object. Contains its own reference to the User node in Firebase for read/write operations |  |
| Court | Data model for the Court object. Contains its own reference to the Court node in Firebase for read/write operations |  |

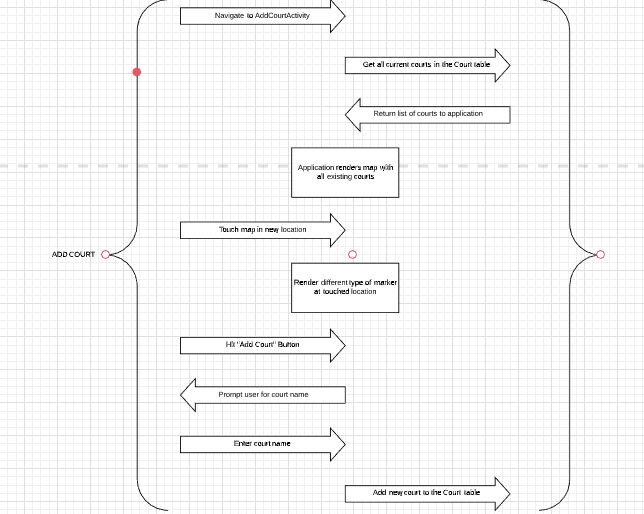
## Use Cases Diagram

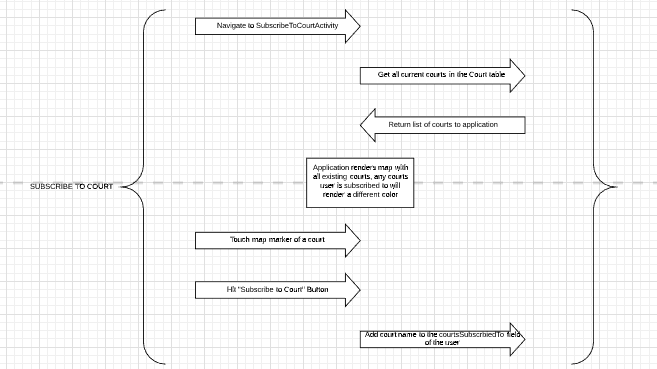


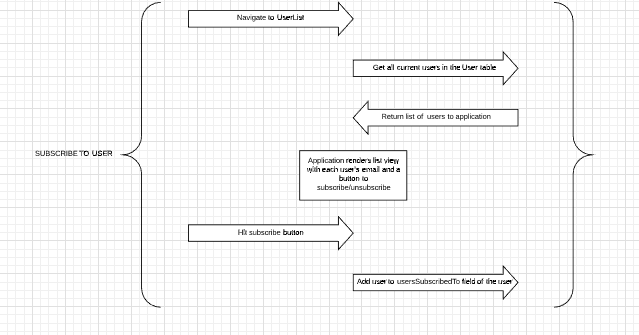
## Sequence Diagram



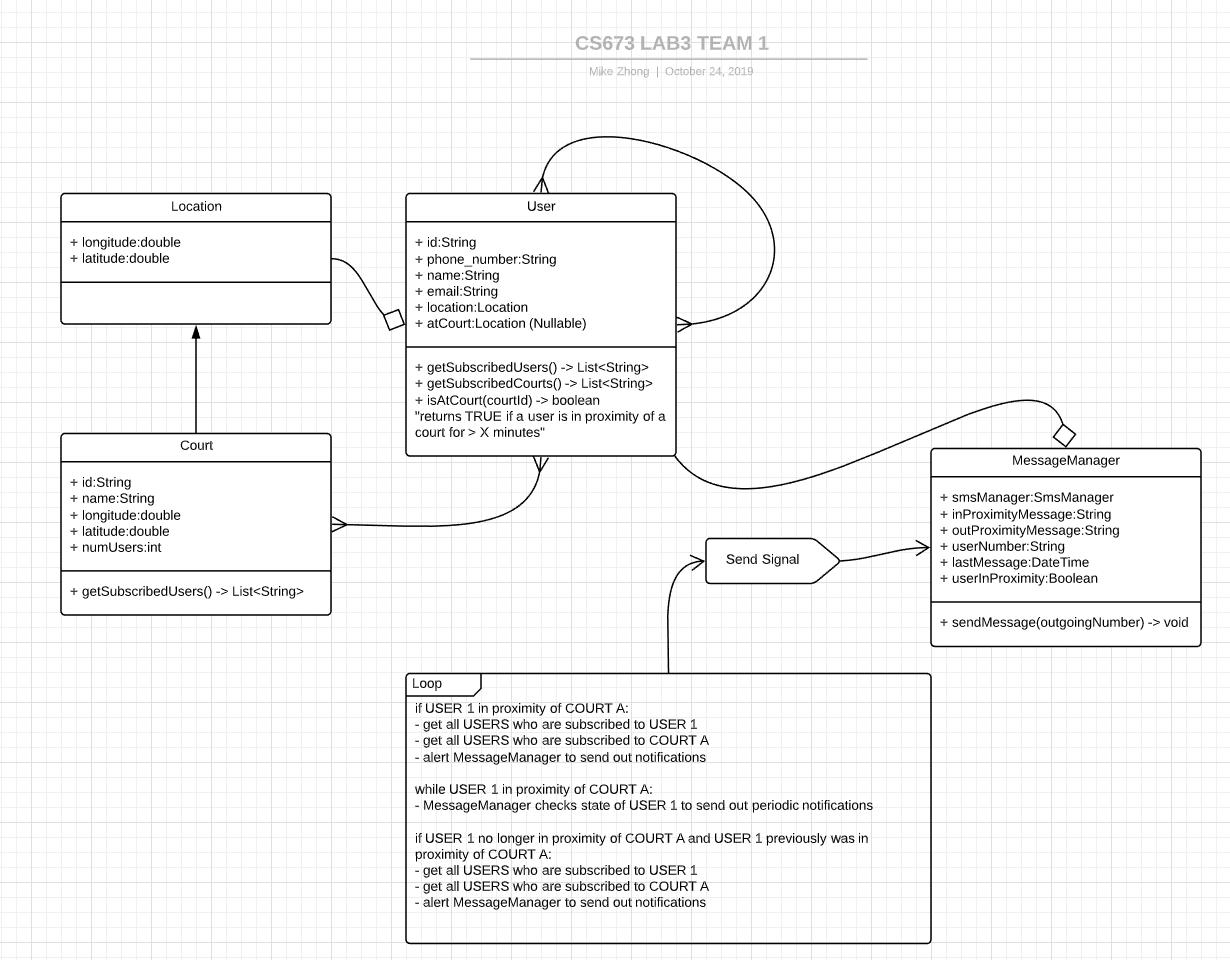






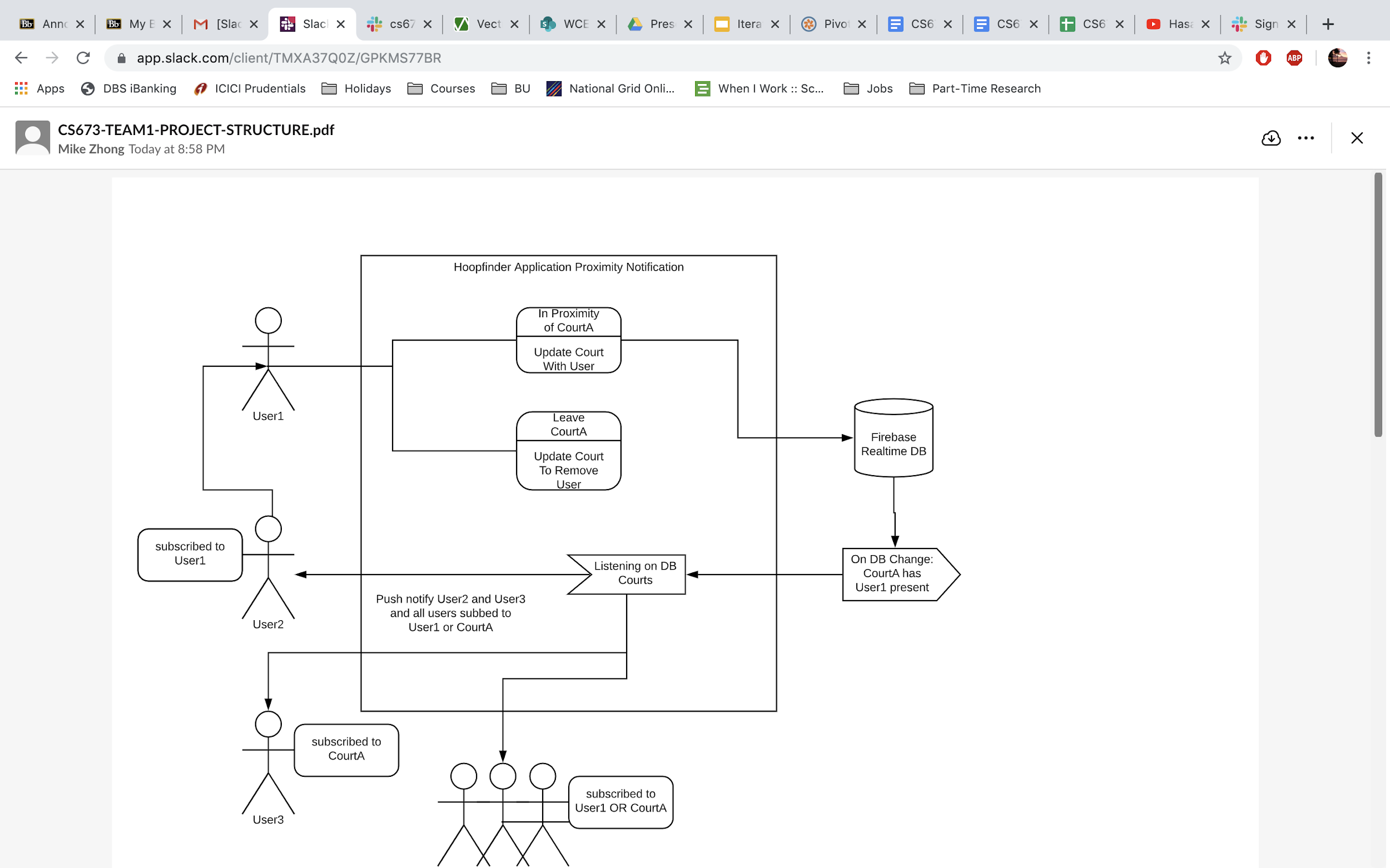


## Class Diagram



# Design Patterns

The below diagram describes the project structure and the flow of the events in the app.



We utilize the Observer/Observable pattern.

Subscribe to Users::

* In effect, the application will need to display views of all users, giving the main user the ability to subscribe or unsubscribe to each user. This view observes the Firebase “User” table for updates to child objects. In this case, when a new user joins, that user will trigger the UI to include that new user in the view.
* When the user subscribes to a user, that results in a change to the current users entry in the “User” table. This view also needs to observe that node to update the text on the button from “subscribe” to “unsubscribe” and vice-versa
* The view is populated by observing these two tables

Add Court:

* For the AddCourtActivity view, the view needs to observe only the “Court” table and render map markers for each court in the table.
* After adding a court by map touch -> button press, the “Court” table will be updated and the new court you added will display with a marker

Subscribe to Court:

* This view needs to observe the “Court” table but also the current user’s node in the “User” table
* All courts are rendered on the map as markers but the map marker will change based on the status (subscribed or not) of the current user.
* If a current user touches a marker -> hit subscribe, that will update the user’s node and the marker will need to render it differently to reflect its status update
* New courts that get added will also trigger the UI to display new markers as new courts are added

Proximity Notification:

* Proximity notifications are implemented as service and observe the “Court” table and “User” table
* No UI, therefore no UI updates
* As location updates are received, proximity checking and timing are executed to determine when a user is “at the court”
* That triggers an update to the specific court node of the “Court” table and the current user is added.
* This ultimately triggers notifications which are sent to users based on user subscription and court subscription status

# Key Algorithms

No advanced algorithms were implemented for this software.

# Classes and Methods

Not applicable

# References

<http://robotics.ee.uwa.edu.au/courses/design/examples/example_design.pdf>

<https://developer.android.com/topic/libraries/support-library>

<https://arxiv.org/ftp/arxiv/papers/1005/1005.0595.pdf>

# Glossary

Not applicable

1. <http://robotics.ee.uwa.edu.au/courses/design/examples/example_design.pdf> [↑](#footnote-ref-0)
2. <https://developer.android.com/guide/platform> [↑](#footnote-ref-1)