1. Project Title and Authors
    a. Your assigned team number (from 1 to 76)
    b. Optionally a cool name you select for your team c. A list of all team members (names, emails, and USC ID numbers)
    Task Breakdown:

    * Feature 1: implemented by everyone working together

    * Feature 2: implemented by everyone working together

    * Feature 3: implemented by everyone working together

    * Basic features: implemented by everyone working together

2. Preface
    a. Describe this document in a nut-shell, the intended readers, etc.

The USCRecApp is an Android application designed for USC students to make reservations on USC recreational centers online. This document provides implementation documentations for our detailed implementation for the USCRecApp. This document is intended for both programmers and clints for reference purposes about the implementation.

3. Introduction
    a. Summarize the content of this document

The USCRecApp allows USC students to make reservations before accessing recreational centers under USC's guidance. Students should be able to view each USC recreational center on the map. Students should be able to make reservations for an available time slot or join a waiting list if the time slot is full. Students should be able to track their reservation information and cancel an upcoming reservation. The USCRecApp would be more user-friendly for mobile users than the current web reservation system.

This document serves as the implementation document for the implementation of this app with two primary purposes. First, it records the changes we made during implementation (on architectural design, detailed class design, or requirements). Second, it provides a rationale behind each of these changes.

4. Architectural Design Change
    a. What changes did you make during the implementation?
    b. The rationale behind these changes

The main Architectural Design style did not change, still using a layered structure.

In the User Interface Layer, we added two components, and changed the name of some components, as shown in the new Architectural design diagram.

1. We added a Main Activity as the entry point to the app (that will be used to check if the user is currently logged in, and based on that information, we will redirect the user to the Login Activity or the Map Activity, etc.)
2. We added a Wifi Activity to deal with the case where the emulator cannot connect to the local wifi and need a cold boot. Since wifi is required for our app, this page is important because it tells the user where the problem is if it cannot be connected to wifi, which sometimes happens as a result of Android Studio's emulator.

In the Application Function Layer, we added one more component: "Receive Notification." We ignored this component previously because we think this can be easily handled when we deal with cancel (because notification is only triggered when someone cancels a booking). However later during implementation we noticed that receiving notification is significantly more complicated than we have imagined. Thus, we isolated this functionality as an individual component.

In the Database Layer, we added two more tables, and changed the name of some components, as shown in the new Architectural design diagram.

1. The first is the DateList table. This table is designed for the convenience of recording reservations. We need to know the date of reservation to divide them into past and upcoming reservations, this table just assigns an id to each date in a year.
2. The second is the availability table. The table is designed to record the availability of a user to make reservations on one specific date. A user is only allowed to make one reservation for each date, so we need to check the availability of the user on the date he or she chooses before allowing the user to make reservations.
3. The FireBase Notification Table (used to handle notification only) is designed to notify users for available slots in the waitlist using FCM. When a user needs to be notified, the system inserts the user to the table. This user listens for the change in the database and gets notified immediately.

5. Detailed Design Change
   a. What changes did you make during the implementation?
   b. The rationale behind these changes

1. Deleted ActivitySwitchHelper Interface -> all handled within each activity for convenience
2. Deleted Waitlist & RecreationCenter -> we noticed that the information we need from Waitlist and RecreationCenter is all already stored in our database, so we don't need a class to hold them, but we can access them directly from the database. After we query

the database, waitlist & recreationCenter appears as a form of data structure available for us to use when we need them only, instead of predefining all of them when starting the app, which might lead to a slower performance.

3. We added UserLogout Interface & Class to handle user logout events, which is an extra functionality not required according to the requirement document, so we did not have these contents in our original detailed design.
4. Also, we created an interface for each of our original classes, which allows extending functionalities in the future easily, making the design more flexible.
5. All other changes are minor (does not affect the overall structure of our design), such as adding/deleting some class attributes to our original class design to make the implementation more convenient, or adding some helper functions for a class aiming to help implement its main methods and make the code cleaner.

6. Requirements Change

a. If you decide to make changes to certain requirements, please document the changes in this section. The changed requirements are not restricted to the ones you implemented for this assignment; you may implement some of them in a later assignment.

b. For each changed requirement, please answer the following questions:

i. Does it change your design?

ii. If not, why not?

iii. If yes, what should be changed and how? Your analysis of the impact on your design will be taken into account again when grading the future assignments.

1. We added a logout functionality in the Profile Page that allows the user to logout of their account and be redirected to the Login Page. This change has minor effects on our main design, we simply added a button and an event handler in the Profile Activity for the logout event.
2. For the functionality of making reservations. We added a check of the user's availability on the specific date. Since a user should only be allowed to make one reservation each day, we need to check if the user already has a reservation on that date before allowing the user to successfully make the reservation. This change does change the design. We added a table in the database to record the availability of the users on different dates.
3. For the waitlist and notification functionality, we allow the user to receive multiple notifications by keeping them in the waitlist until either the time passed or the user joined the corresponding timeslot. By this implementation, if the user received notification but another user took the newly opened space, the user doesn't need to join the waitlist again, and will continue receiving notifications until the user finally makes the booking successfully. It does not change the design. What we do is to check if the user is in the waitlist when the user makes a reservation. If yes, then delete it from the waitlist. Those users who joined the waitlist are still in the waitlist if they do not make a reservation.