Implementation Document

1. **Project Title and Authors:**
    a. Assigned team number: 19
    b. Team Arnold
    c.

Name	USC ID
Luis Contreras (Feature 1)	5287392487
Jadrian Tan (Feature 2)	7122221396
Alexander Rouse (Feature 3)	6995520656

2. **Table of Contents**

3. **Preface**
    a. This is our Spring 2022 CSCI310 group project. We have been assigned Project C, USCRecApp. Our app will allow users to book and modify USC recreational center reservations.
4. **Introduction**
    a. USC requires users to create reservations before using the gym facilities. USCRecApp aims to solve this problem by providing an easy to use system for students to make reservations online. The app will be interacting with the database USC uses to track bookings. USCRecApp aims to simplify and organize the reservation process on behalf of USC.
5. **Glossary**
    a. USC - University of Southern California
    b. USCRecApp - Name of the application that this document describes.
    c. Team Arnold - Team of qualified software engineers who worked to create the application that this document documents.
    d. Assigned team number - Number given to us by the commissioner, USC, of this application.
    e. Architectural Design Changes- Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system.
    f. Detailed Design Changes - Detailed design focuses on all of the implementation details necessary to implement the architecture that is specified.
    g. Requirements Change - Changes that are being made to requirements provided.

6. **Architectural Design Changes**
    a. We did not make any changes to the architectural design. We implemented all of the architectural designs that we described on 2.2
    b. We did not make any changes to the architectural design. We implemented all of the architectural designs that we described on 2.2

7. **Detailed Design Changes**
    a. Changes that were made to the implementation are the following:
        i. Since we are coding in an event driven language, instead of calling all of the classes in the main function, we called classes respectively when a certain action was initiated. That meant that the main function only called the Login class and the login class took care of all the other classes that we documented.
        ii. Instead of having a RecCenter class, we replaced it by representing each RecCenter as a collection in our Firestore Database. Aside from that, instead of storing the "days" into a HashMap, each "day" is now represented as a document under <Rec_Center> collections.
        iii. The Time_Slots class was also deleted and got implemented as a "field" in our RecCenter collection in the database. Instead of storing a string for the start time and another string for the end time, we decided to just have one string that included the start time and the end time being represented as a "field," with # of capacity as its "value."
        iv. Feature2.java class was created and implemented that would allow users to check for available bookings for the selected RecCenter and book an appointment (if available) or set a reminder (if not available). To ensure the correct RecCenter availability is being displayed, we used Bundle (a built-in class on Android Studio) to store the name of the RecCenter collection when starting the Feature2 activity. In this class, we handled the buttons and info display under the onCreate() function. We also implemented three separate functions: updateDB(String, String, int), updateBooking(String), and updateReminder(String).
        v. Another class was created: BookingPage. The booking button and the upcomingBookings data members of the Map_Interface were added to this class.
        vi. Lastly, the Login_Page was implemented in the firestore database so we did not need to explicitly create the functions in the Login_Page such as verifyUser(), hash(), and getConnection(). We simply grabbed the relevant information from the GUI and stored that information within the user class in the database.
    b. The rationale behind these changes
        i. We noticed that it made more sense to create certain classes when they would be called instead of leaving them in the main. This helped us better organize our code to have an easier time using certain classes as the user navigated through the different pages.
        ii. For the RecCenter Class and HashMap removal, we discussed and agreed upon the said approach above for better data storage and organization. As the amount of days increases, Firebase will continue to give us a consistent and correct result.
        iii. The Time_Slots class was implemented into the RecCenter class because it was easier for us to retrieve the start and end time if it were stored in the RecCenter class. We decided to have

a single string to represent the start +end time with # of capacity as its value, because it made it easier for us to display this string in the GUI that we created.

iv. Feature2.java was implemented to correctly display the available bookings (for 3 consecutive days) and to properly implement both the Book-btn and Remind_Me-btn. In regards to the three functions said on 7a(iv), updateDB is responsible for updating the capacity count under each <Rec_Center> collection. updateBooking is responsible for updating the "Upcoming_Appt_<i>" field under the "users" collection. Lastly, updateReminder is responsible for updating the "Reminder_<i>" field under the "users" collection.

v. The BookingPage was created because we were displaying all of the booking information on a different page, it was no longer with the Google Map class. This made it easier for Alex (teammate 3) to work on displaying feature 3 without it being connected to the RecCenter.

vi. We noticed that the firestore database had the ability to verify users. Therefore, we decided not to code these functions as they were implemented for us in the firestore database. In addition, we were not able to store a Person class into the database. Thus, we were forced to store key value pairs in the user collection of our database

8. **Requirements Changes**

a. Regarding the design of feature 2, our customer, Yannan and us discussed and agreed upon the design of the UI from "tabbed activity + fragments" to a single-activity-style UI on Android Studio.

b. No other changes were made to the requirements set by the customer, Yannan.

9. **Revised UML diagram**