

Software Requirements Specification

**Document**

**Soccer Management App: Soccer Alliance**

# Guided By:

**Prof. Olivier Melançon**

**Prof. Sakshi Sharma**

**Prof. Harsh Divecha**

**Prof. Navpreet Kaur**

|  |  |  |
| --- | --- | --- |
|  | **Prepared By:** |  |
| Yash Modi | - | 1894082 |
| Parag Dangariya | - | 1894264 |
| Monika Upadhyay | - | 1894220 |
| Ami Parekh | - | 1894192 |
| Farhin Vahora | - | 1894132 |
| Saurabh Jain | - | 1894017 |

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| 1 | [Introduction](#Introduction) | [3](#Introduction) |
| 1.1 | [Purpose](#Purpose) | [3](#Purpose) |
| 1.2 | [Scope](#Scope) | [3](#Scope) |
| 2 | [General Description](#General_Description) | [3](#General_Description) |
| 2.1 | [Product Functions](#Product_Functions) | [3](#Product_Functions) |
| 2.2 | [User Characteristics](#User_Char) | [4](#User_Char) |
| 3 | [Specific Requirements](#Specific_Req) | [5](#Specific_Req) |
| 3.1 | [Functional Requirements](#Functional_Req) | [5](#Functional_Req) |
| 3.2 | [Non-Function Requirements](#NFR) | [7](#NFR) |
| 4 | [Analysis Models](#Analysis_Models) | [8](#Analysis_Models) |
| 4.1 | [Use-Case Diagram](#Use_case) | [8](#Use_case) |
| 4.2 | [Class Diagram](#Class_Diagram) | [9](#Class_Diagram) |
| 4.3 | [Scenarios](#Scenarios) | [10](#Scenarios) |
| 5 | [Database](#db) | [38](#db) |
| 5.1 | [Database Schema](#db_schema) | [38](#db_schema) |
| 5.2 | [Database Credentials](#_Database_Credentials) | [39](#_Database_Credentials) |
| 5.3 | [Database Script](#db_script) | [39](#db_script) |
| 6 | [Database Web Services’ Calls Details](#webcalls) | [42](#webcalls) |
| 7 | [Prototype](#prototype) | [60](#prototype) |
| 8 | [References](#_References) | [68](#_References) |

**Table of Contents**

# Introduction

This document depicts the software requirements for a mobile application referred to as soccer management app. It provides the general description of the project, it's functions, a list of application attributes, database requirements, interface, performance, use case modeling diagrams and GUI design diagrams.

# Purpose

The main purpose of this soccer management application is to manage the leagues, teams, players that participate in the league, create schedules for the matches, enter the result of the matches and to display various standings and statistics of the match.

# Scope

The soccer management mobile application is named as “**Soccer Alliance**”. Soccer Alliance provides three types of users: League Manager, Team Manager and Guest User. The ambition of Soccer Alliance is to provide a user to manage league, teams, players participating in the match and also view score of the match.

Internet connection is mandatory for Soccer Alliance. Application requires Internet access to fetch and display result from database server. Advantage of the application is to give a user an in-hand experience that a league manager or team manager experiences in creating, managing, updating the leagues, teams, players, strategies, etc.

# General Description

This section of the document covers an overview of the application. How application works and its basic functions. What features are available for specific user and how database of this application has to be user specific.

* 1. **Product Functions**

**Soccer Alliance** includes following functions.

* **Soccer Alliance** provides limited feature access to its guest users and all premium features to its logged in users.
* The application has login feature which uses Firebase authentication for the user to log in.
* The application also has a provision for the forgot password.
* The application displays the list of Team members of the match with Team Manager.
* Team manager can manage only one team and one team can have only one manager.
* League manager have same rights as team manager but they have access to all teams.
* League manager can enter results of the match.
* League manager can update the score of the match.
* League manager can reschedule or cancel the match.
* League manager can update standings and statistics of the match.

# User Characteristics

**Soccer Alliance** is not restricted to specific users. Application is for all the users who use the mobile applications and who are enthusiast of soccer. The end-users of **Soccer Alliance** are Guest user and logged in users such as League Managers and Team Managers.

**Guest** users are confined to only view leagues, view teams, match schedules, list of players and view the score of the matches.

**League Managers** can create leagues and arrange schedules, add or remove the teams and as well as reschedule or cancel the matches. League Managers also have the authority to display the score of the match.

**Team Managers** can add or remove players from the team. One team can have only one manager and vice-versa. They have the authority to replace the player from the team as per the opponent team and the league. They can edit the information of the players.

# Specific Requirements

This section of the document set out the detailed description of the system and all its features.

# Functional Requirements

This section includes all the requirements that specify all the fundamental actions of the **Soccer Alliance** Application.

|  |  |  |  |
| --- | --- | --- | --- |
| FUNCTIONAL REQUIREMENTS | | | |
| Sr. No. | **Name** | **Description** | **Must/Could** |
| FR-1 | Sign Up | An application must allow its users to register themselves. Sign up requires user’s full-name, age, gender, country, email, phone number, password and confirm password. | Must |
| FR-2 | Login | **Soccer Alliance** must have login feature which takes user’s email and password. Login allows user to access all the premium features of the application.  **Soccer Alliance** have two users who can log-in, i.e., Team manager and League manager who have overall access to the application. | Must |
| FR-3 | Forgot Password | **Soccer Alliance** must have a feature to allow its users to reset the password. Forgot password activity ask for user’s email. A link is sent to user’s email to reset password. After resetting, firebase automatically updates the password. | Must |
| FR-4 | View List of Leagues | Guest and logged-in users can view the list of leagues. | Could |
| FR-5 | View List of Teams | Guest users can view the list of teams. | Could |
| FR-6 | View Score | Guest and logged-in user can view the score of the match. | Could |
| FR-7 | View Upcoming & Played Matches | Guest can view the list of upcoming and played matches. | Could |
| FR-8 | Add/Remove Team into League | League Manager can add or remove teams in the league. | Could |
| FR-9 | Add/Remove Team Players | Team Manager are able to add and remove team players. Adding of a new team player includes player profile picture, player name and position. | Could |
| FR-10 | Create League & Schedules | League managers have right to create new leagues and schedules of that league. The league managers also have the authority to create leagues by adding matches for the teams. They are authorized to make schedules for a particular league by adding teams, day, time, date and location of the match. | Must |
| FR-11 | Edit Team Information | Team manager can edit player’s information. Editing a player’s information includes all the details which were provided at the time of adding team player. | Could |
| FR-12 | Display Score of the match | **Soccer Alliance** displays the score of the recent matches that has been played. It shows the number of goals that each team scored for the match. | Must |
| FR-13 | Display Standings & Statistics | **Soccer Alliance** also displays the standings & statistics which includes shots, shots on target, possession, passes, pass accuracy, fouls, yellow cards, red cards, offside and corners for each team. | Must |
| FR-14 | Display Line-Ups | **Soccer Alliance** displays the position of each player in the team. | Must |

# Non-Functional Requirements

This section of document specifies how the system should behave and quality characteristics of **Soccer Alliance.**

|  |  |  |
| --- | --- | --- |
| NON-FUNCTIONAL REQUIREMENTS | | |
| Sr. No. | **Name** | **Description** |
| NFR-1 | Performance Requirements | The application should be fast and robust when loading. The application should not take more than 5 seconds to respond. |
| NFR-2 | Usability Requirements | The application should be easy to use and intuitive. The application should have a user-friendly interface. GUI should be simple and clear. |
| NFR-3 | Reliability Requirements | The application must give an error message in case of error. |
| NFR-4 | Delivery Requirements | The application must be delivered no later than 14th, March. |
| NFR-5 | Implementation Requirements | The application must be implemented using Android 5.1+ Versions, MySQL, Clever Cloud Platform and Firebase Authentication. |
| NFR-6 | Security Requirements | The application should not allow unauthorized attempts at usage. The application should protect the user authentication using email verification. |
| NFR-7 | Availability Requirements | The application must meet or exceed 99% uptime. The application downtime shall not exceed 3.65 day/year (14.4 minutes per day). |
| NFR-8 | Inter-Operability Requirements | The application should allow access to the different sections of the application without altering its efficiency and consistency. |
| NFR-9 | Privacy Requirements | The personal information of the registered user should only be accessed by the user itself and the administrator. |

# Analysis Models

This section of the document defines the structural diagram and flow of the **Soccer Alliance** application.

# Use Case Diagram

# F:\MAD\Semester-4\Soccer management app\Structural Diagrams\SA-UseCase.jpg

# Class Diagram

# 

# 

# Scenarios

|  |  |
| --- | --- |
| List of Use-Cases | |
| UC1 – Login  UC2 – Sign Up  UC3 – Edit Profile  UC4 – Forgot Password  UC5 – View Leagues  UC6 – View Upcoming Matches  UC7 – Create Team  UC8 – Edit Team Information  UC9 – Add Player  UC10 – Delete Player | UC11 – Create League  UC12 – Schedule Match  UC13 – Reschedule Match  UC14 – Add Score  UC15 – Update Score  UC16 – View Teams  UC17 – Add Team to League  UC18 – Remove Team from League  UC19 – Cancel Match |

1. **Login**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-1 | |
| Use Case Name | Login | |
| Date created | 24/01/2020 | |
| Actor | Registered user | |
| Description | When user wants to access their account and wants to create and manage teams, create leagues, make schedule and add or remove players, for that necessary to do login first. | |
| Triggering event | User enters user Email ID and password to access the whole system. | |
| Preconditions | User must have an account. | |
| Flow of events | Actor | System |
| 1. Enter email and password. 2. Click on Login Button. | * 1. Check email and password in database.   2. Redirect User to dashboard screen. |
| Alternative flow of events | Actor | System |
| 1. Enter email and password. 2. Click on Login Button. 3. Repeat step 1 and 2. | * 1. Verification of email and password is done in Firebase Authentication.   2. Gives error message to user that, “Email or password you have entered is incorrect!”   3. Clear text of email and password field for user to enter valid email and password. |
| Post conditions | User have now logged in into system. | |
| Exception - Conditions | User does not have an account, then user have to create the new account by clicking register now. | |

1. **SignUp**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-2 | |
| Use Case Name | Signup | |
| Date created | 24/01/2020 | |
| Actor | Team manager and League manager | |
| Description | Sign up activity contains user’s name, email, phone number, user type, password, confirm password, full name, age, gender and country. | |
| Triggering event | User clicks on the Sign-up button from register page. | |
| Preconditions | User does not have an account. | |
| Flow of events | Actor | System |
| 1. Enter email address. 2. Enter phone number. 3. Select user-type. 4. Enter name, age, gender, country. 5. Create new password and confirm password. 6. Click on next Button. 7. Select link sent on user’s email. | 1.1 Check email that does not exist in Firebase.  5.1 Check Password criteria is correct.  5.2 Matches password with confirm password.  6.1 A link is sent on user’s email.  6.1 User is verified by Firebase. |
| Alternative flow of events | Actor | System |
| 1. Enter email address, phone number and user-type. | * 1. Check if email exist in database.   2. Notify User with   message, “User |
|  | 1. Repeat step 1. 2. Enter name, age, gender, and country. 3. Enter valid password. 4. Repeat step 4. 5. Confirm the password. 6. Repeat step 6. 7. Click on next Button. 8. Select link. 9. Firebase verifies the user. | Already exists please enter different email.”   * 1. Clear text of email field.   3.1 Check all the required information is not null.  3.2 Notify User with message, “All fields are required, enter the missing details.”   * 1. Check Password length is correct.   2. Notify User with message, “Password does not match to requirements”.   3. Clear text of password field.   4. Matches password with confirm password.   5. Notify User with message, “Password and confirm password must be same.”   6. Clear text of confirm password field.   8.1 Send link on user’s   email. |

|  |  |
| --- | --- |
| Post conditions | User now have an account. |
| Exception - Conditions | User must enter unique email. |

1. **Edit Profile**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-3 | |
| Use Case Name | Edit Profile | |
| Date created | 24/01/2020 | |
| Actor | Logged in user | |
| Description | Edit Profile activity allows user to change profile picture, user’s name, age, gender, country and phone number. | |
| Triggering event | User clicks on the my profile button from drawer menu. | |
| Preconditions | User is logged-in into system. | |
| Flow of events | Actor | System |
| 1. Select my profile from drawer menu in dashboard screen. 2. User modifies the information. 3. Click on update details button. | 1.1 Display user’s personal information.   * 1. Check if user has modified any information.   2. Update user’s information in database.   3. Notify user with message, “Profile updated”. |
| Alternative flow of events | Actor | System |
| 1. Select my profile from drawer menu in dashboard screen. 2. User modifies information. 3. Click on update details button.     4. Repeat step 2 and 3. | 1.1 Display user’s personal information.   * 1. Check User have modified any information.   2. Notify user with message, “To update   profile, the information must have been changed.”. |
| Post conditions | User have changed his/her personal information. | |
| Exception - Conditions | System displays the user’s current information on screen. | |

1. **Forgot Password**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-4 | |
| Use Case Name | Forgot Password | |
| Date created | 24/01/2020 | |
| Actor | Registered user | |
| Description | Forgot password activity allows user to reset their password. | |
| Triggering event | User clicks on the forgot password button from login screen. | |
| Preconditions | User must has an account. | |
| Flow of events | Actor | System |
| 1. Click on forgot password button from login screen. 2. Enter the email address. 3. Click on next Button. 4. Enters new password. | 1.1 Display the screen for entering email address.   * 1. Check email exists in database.   2. New password link is sent to user’s email.   3. Firebase directs to new password webpage.   4. Firebase validates password.   5. Save the password into Firebase.   6. Redirect user to login page. |
| Alternative flow of events | Actor | System |
| 1. Click on forgot password button from login screen. 2. Enter the email address. 3. Click on next button. 4. Repeat step 2 and 3. 5. User enters new password. 6. User clicks on next button to save new password. | 1.1 Display the screen of entering email address.   * 1. Check if email exists in database.   2. Notify user with message, “User does not exist. Please enter different email.”.   3. Clear text of email field.   4. Firebase validates password. |
| Post conditions | User can reset the password. | |
| Exception - Conditions | User does not have an account, then redirect user to registration page. | |

1. **View Leagues**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-5 | |
| Use Case Name | View leagues | |
| Date created | 24/01/2020 | |
| Actor | All user | |
| Description | View leagues allows user to see list of leagues. | |
| Triggering event | User clicks on particular league to explore details about league from dashboard screen. | |
| Preconditions |  | |
| Flow of events | Actor | System |
| 1. Click on league from dashboard screen. 2. Click on particular league from the list  to view teams of league. 3. User selects team from the list to view players of that team. | 1.1 Display a list of leagues from database.  2.1 Display a list of teams in particular league from database.  3.1 Display the list of players from the database. |
| Alternative flow of events | Actor | System |
| 1. Click on league from dashboard screen. 2. Click on particular league from the list  to view teams of league. 3. User selects team from the list to view players of that team. | 1.1 Display a list of leagues from database.  3.1 Notify User with   message, “Could not   display player list,   please try again”. |
| Post conditions | User sees all the available leagues stored in database. | |
| Exception - Conditions |  | |

1. **View Upcoming Matches**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-6 | |
| Use Case Name | View upcoming matches | |
| Date created | 24/01/2020 | |
| Actor | Logged user and guest user | |
| Description | View upcoming matches allows user to see list of upcoming   matches. | |
| Triggering event | User clicks on particular match to explore details about match from   dashboard screen. | |
| Preconditions |  | |
| Flow of events | Actor | System |
| 1. Click on matches tab from dashboard screen. 2. Click on particular match from upcoming match list to view details of match. | 1.1 Display a list of upcoming matches from database.  2.1 Display the match information’s like players list, team manager’s name and league name from database. |
| Alternative flow of events | Actor | System |
| 1. Click on match from dashboard screen. 2. Click on particular match from upcoming match list to view details of match. | 1.1 Display a list of matches from database.  2.1 Notify User with   message, “Could not   display match details,   please try again”. |
| Post conditions | User sees all the available match details in database. | |
| Exception - Conditions |  | |

1. **Create Team**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-7 | |
| Use Case Name | Create Team | |
| Date created | 24/01/2020 | |
| Actor | Team Manager | |
| Description | Create team activity allows user to create their own team in system. | |
| Triggering event | User selects’ the my team from drawer menu. | |
| Preconditions | User is logged in into system. | |
| Flow of events | Actor | System |
| 1. Select my team from drawer menu. 2. Enter the name of team. 3. Upload the logo of team. 4. Enter the team shorthand. 5. Click on the update team details button. | 1.1 Display my team screen.   * 1. Verify all Details.   2. Notify user with message, “team added successfully”. |

|  |  |  |
| --- | --- | --- |
| Alternative flow of events | Actor | System |
| 1. Select my team from drawer menu. 2. Enter the name of team. 3. Upload the logo of team. 4. Enter the team shorthand. 5. Click on the update team details button. | 1.1 Display create team screen.   * 1. Verify all entered details.   2. Notify user with message “Please enter all required information”. |
| Post conditions | User have added Team. | |
| Exception - Conditions | User is not allowed to add team without team logo, team shorthand and name of the team. | |

1. **Edit Team Information**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-9 | |
| Use Case Name | Edit Team information | |
| Date created | 24/01/2020 | |
| Actor | Team manager | |
| Description | Edit team information activity allows user to Edit their own team information on system. | |
| Triggering event | User clicks on the team button. | |
| Preconditions | User have at least one added team. | |
| Flow of events | Actor | System |
| 1. Select my team from drawer menu. 2. User modifies information. 3. Click on the Update team details button. | * 1. Display my team screen.   2. Notify user with message, “team details modify successfully”. |
| Alternative flow of events | Actor | System |
| 1. Select my team from drawer menu. 2. User modifies information. 3. Click on the Update team details button. | 1.1 Display edit team screen.  3.1 Notify user   with message   “Please enter   all required   information”. |
| Post conditions | User have updated team information. | |
| Exception - Conditions | User is not allowed to add team without team logo, team label and name of the team. | |

**9. Add Player**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-10 | |
| Use Case Name | Add player | |
| Date created | 24/01/2020 | |
| Actor | Logged user | |
| Description | Add player activity allows user to add player in team. | |
| Triggering event | User clicks on the add player button from player list. | |
| Preconditions | User should be logged-in to system. | |
| Flow of events | Actor | System |
| 1. Click on the add player button from player list screen. 2. Enter the name of player. 3. Upload the photo of the player. 4. Enter the role of player. 5. Enter the strength of the player. 6. Click on the add player button. | 1.1 Display player list screen.   * 1. Verify all details entered.   2. Notify user with message “Please enter all required information”. |
| Alternative flow of events | Actor | System |
| 1. Click on the add player button from player list screen. 2. Enter the name of player. 3. Upload the photo of the player. 4. Enter the role of player. 5. Enter the strength of the player. 6. Click on the add player button. | 1.1 Display player list screen.   * 1. Verify all details entered.   2. Notify user with message “Please enter all required information”. |
| Post conditions | User have added player. | |
| Exception - Conditions | User is not allowed to add player without name, photo and role of the player. | |

**10.** **Delete Player**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-11 | |
| Use Case Name | Delete Player | |
| Date created | 24/01/2020 | |
| Actor | Logged user | |
| Description | Delete player activity allows user to delete player from team and also from database. | |
| Triggering event | User clicks on the delete player button. | |
| Preconditions | User have at least one player. | |
| Flow of events | Actor | System |
| 1. Click on remove player from added player details screen. | * 1. Delete player from database. |
| Alternative flow of events | Actor | System |
| 1. Click on remove player from added player details screen. | 1.1 Delete player from database. |
| Post conditions | User have deleted the player. | |
| Exception - Conditions | User is not able to delete player without adding any player. | |

1. **Create League**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-12 | |
| Use Case Name | Create League | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Create League allows user to create league on system. | |
| Triggering event | User clicks on the Create League button in League List screen. | |
| Preconditions | User is logged in into system. | |
| Flow of events | Actor | System |
| 1. Click on the Create League button. 2. Upload the logo of league. 3. Enter the name of league. 4. Enter the no. of teams in league. 5. Click on the Create New League button. | 1.1 Display Create League screen.     * 1. Verify all Details.   2. Display league list screen. |

|  |  |  |
| --- | --- | --- |
| Alternative flow of events | Actor | System |
| 1. Click on the Create League button. 2. Upload the logo of league. 3. Enter the name of league. 4. Enter the no. of teams in league. 5. Click on the Create League button. 6. Repeat step 1 to 5. | 1.1 Display Create league screen.  5.1 Notify user with message “Please enter all required information”. |
| Post conditions | User have added League. | |
| Exception - Conditions | User is not allowed to add league without league name, logo and no. of teams in league. | |

1. **Schedule Match**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-13 | |
| Use Case Name | Schedule match | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Schedule match allows user to Schedule the matches for league. | |
| Triggering event | User clicks on the Schedule match button from league operation screen. | |
| Preconditions | User is logged in into system. | |
| Flow of events | Actor | System |
| 1. Click on the Schedule match button from league operations screen. 2. Enter team1 name. 3. Enter team2 name. 4. Select date for match 5. Select time for match. 6. Enter location for match. 7. Click on the Schedule   Match button. | 1.1 Display Schedule match screen.     * 1. Verify all Details.   7.2 Notify user with  Message “match   Scheduled   Successfully”. |
| Alternative flow of events | Actor | System |
| 1. Click on the Schedule match button from league operations screen. 2. Enter team1 name. 3. Enter team2 name. 4. Select date for match 5. Select time for match. 6. Enter location for match. 7. Click on the Schedule   Match button. | 1.1 Display Create league screen.  7.1 Verify all entered   details.  7.2 Notify user with  message “Please  enter all required  information”. |
| Post conditions | User have scheduled match for league. | |
| Exception - Conditions | User is not allowed to schedule match without team’s names, date, time and location of match. | |

1. **Reschedule Match**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-14 | |
| Use Case Name | Reschedule match | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Reschedule match allows user to Reschedule a matches for league. | |
| Triggering event | User clicks on the Reschedule match button from upcoming match. | |
| Preconditions | User is logged in into system and user already schedule at least one match. | |
| Flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Select Upcoming matches. 4. Select match. 5. Select reschedule match. 6. Select date for match. 7. Select time for match. 8. Enter location for match. 9. Click on the Reschedule   Match button. | 1.1 Display League List.    9.1 Verify all Details.  9.2 Notify user with  Message “match   Scheduled   Successfully”. |
| Alternative flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Select Upcoming matches. 4. Select match. 5. Select reschedule match. 6. Select date for match. 7. Select time for match. 8. Enter location for match. 9. Click on the Reschedule   Match button. | 1.1 Display League List.    9.1 Verify all Details.  9.2 Notify user with  Message “match   Scheduled   Successfully”. |
| Post conditions | User have Rescheduled match for league. | |
| Exception - Conditions | User is not allowed to schedule match without date, time and location of match. | |

1. **Add Score**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-15 | |
| Use Case Name | Add Score | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Add Score activity allows user to add score of match. | |
| Triggering event | User clicks on the add score button from match’s dashboard screen. | |
| Preconditions | User is logged in into system and user at least added one match. | |
| Flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Click on Played matches tab. 4. Enter the no. of goals. 5. Enter the no. of shots. 6. Enter the no. of shots on target. 7. Enter the no. of possession. 8. Enter the no. of passes. 9. Enter the no. of pass accuracy. 10. Enter the no. of fouls. 11. Enter the no. of yellow cards. 12. Enter the no. of red cards. 13. Enter the no. of off-sides. 14. Enter the no. of Corners. 15. Click on the add score button. | 1.1 Display Match score screen.  15.1 Verify all Details.  15.2 Notify user with  message, “Score added  successfully”.    15.3 Redirect user to   match’s main screen. |
| Alternative flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Click on Upcoming matches tab. 4. Enter the no. of goals. 5. Enter the no. of shots. 6. Enter the no. of shots on target. 7. Enter the no. of possession. 8. Enter the no. of passes. 9. Enter the no. of pass accuracy. 10. Enter the no. of fouls. 11. Enter the no. of yellow cards. 12. Enter the no. of red cards. 13. Enter the no. of off-sides. 14. Enter the no. of Corners. 15. Click on the add score button. | 1.1 Display match score screen.   * 1. Verify all   entered details.   2. Notify user with   message “Please   enter all required   information”. |
| Post conditions | User have added score of match. | |
| Exception - Conditions | User is not allowed to add score without no. of goals, no. of pass and no. of shots in match. | |

1. **Update Score**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-16 | |
| Use Case Name | Update Score | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Update Score activity allows user to update score of match. | |
| Triggering event | User clicks on the update score button from match’s dashboard screen. | |
| Preconditions | User is logged in into system and at least one score should be pre-added. | |
| Flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Click on Played matches tab. 4. Enter the no. of goals. 5. Enter the no. of shots. 6. Enter the no. of shots on target. 7. Enter the no. of possession. 8. Enter the no. of passes. 9. Enter the no. of pass accuracy. 10. Enter the no. of fouls. 11. Enter the no. of yellow cards. 12. Enter the no. of red cards. 13. Enter the no. of off-sides. 14. Enter the no. of Corners. 15. Click on the update score button. | 1.1 Display Match score screen.    15.1 Verify all Details.  15.2 Notify user with  message, “Score  updated successfully”.  15.3 Redirect user to   match’s main screen. |
| Alternative flow of events | Actor | System |
| 1. Select My League from drawer menu. 2. Select league from league list. 3. Click on Upcoming matches tab. 4. Enter the no. of goals. 5. Enter the no. of shots. 6. Enter the no. of shots on target. 7. Enter the no. of possession. 8. Enter the no. of passes. 9. Enter the no. of pass accuracy. 10. Enter the no. of fouls. 11. Enter the no. of yellow cards. 12. Enter the no. of red cards. 13. Enter the no. of off-sides. 14. Enter the no. of Corners. 15. Click on the update score button. | 1.1 Display match score screen.   * 1. Display upcoming match list.      * 1. Verify all   entered details.   2. Notify user with   message “Please   enter all   required   information”. |
| Post conditions | User have updated score of match. | |
| Exception - Conditions | User is not allowed to update score without no. of goals, no. of pass and no. of shots in match. | |

1. **View Teams**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-17 | |
| Use Case Name | View Teams | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | View Teams activity allows user to see all the teams on system. | |
| Triggering event | User should select team list from drawer menu. | |
| Preconditions |  | |
| Flow of e vents | Actor | System |
| 1. Select team list from drawer menu. 2. User view all teams. 3. User can select any particular team for the list of players. | 1.1 Display a list of teams from database.  3.1 Display the players of that particular team fetched from database. |
| Alternative flow of events | Actor | System |
| 1. Select team list from drawer menu. 2. User view all teams. 3. User selects any particular team for details. 4. Repeat step 1. | 1.1 Display a list of teams from database.  3.1 Notify User with message, “Could not load details please try again”. |
| Post conditions | User sees all the available teams in database. | |
| Exception - Conditions |  | |

1. **Add Team to League**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-18 | |
| Use Case Name | Add Team to league | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Add Team to league allows user to add team in league. | |
| Triggering event | User clicks on the Add team button from league operations screen. | |
| Preconditions | User is logged in into system and user created at least on league. | |
| Flow of events | Actor | System |
| 1. User clicks on the Add team button from league operations screen. 2. Select teams from the list.      1. Click on the Add team into league button. | * 1. Display Add teams into league Screen.   2. Display list of teams from database.      * 1. Verify all Details.   2. Notify user with message, “team added successfully”. |
| Alternative flow of events | Actor | System |
| 1. User clicks on the Add team button from league’s main screen. | * 1. Display Add teams into league Screen.   2. Display message “Could not display list of teams from database”. |
| Post conditions | User have added Team into league. | |
| Exception - Conditions |  | |

1. **Remove Team from League**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-19 | |
| Use Case Name | Remove team from league | |
| Date created | 24/01/2020 | |
| Actor | League Manager | |
| Description | Remove team from league activity allows user to delete team from  System. | |
| Triggering event | User clicks on the Remove team button. | |
| Preconditions | User have at least one team. | |
| Flow of events | Actor | System |
| 1. Click on the remove button from added team details screen. 2. Select teams from list. 3. Click on Remove team from league button. | 1.1 Display list of teams added before in that league.  3.1 Display the pop-up box with message “This team will no longer be available.” and “Continue” or “Cancel” button. |
| Alternative flow of events | Actor | System |
| 1. Click on the remove button from added team details screen. 2. Select teams from list. 3. Click on Remove team from league button. | 1.1 Display list of teams from that league.  3.1 Display the pop-up box with message “This team will no longer be available.” and “Continue” or “Cancel” button. |
| Post conditions | User have deleted the team from league. | |
| Exception - Conditions | User is not able to delete team without adding any team into league. | |

1. **Cancel Match**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-20 | |
| Use Case Name | Cancel match | |
| Date created | 24/01/2020 | |
| Actor | League manager | |
| Description | Cancel match allows user to Schedule a matches for league. | |
| Triggering event | User clicks on the Cancel match button from screen. | |
| Preconditions | User is logged in into system. | |
| Flow of events | Actor | System |
| 1. Click on the cancel match button from screen. 2. Click on the continue button from pop-up box. | 1.1 Display the pop-up   box with message   “This match will no   longer be available.”   and “Continue” or   “Cancel” button.  2.1 Delete the team from   league.  2.2 Redirect the user on   the league main page   activity. |
| Alternative flow of events | Actor | System |
| 1. Click on the cancel match button from screen. 2. Click on the cancel button from pop-up box. | 1.1 Display the pop-up box with message “This match will no longer be available.” and “Continue” or “Cancel” button.  2.1 Dismiss the pop-up   box. |
| Post conditions | User have deleted the team from league. | |
| Exception - Conditions | User is not able to delete team without adding any team into league. | |

# Database

This section of the document specifies the database structure with all the entities and their attribute. It also specifies the relationship between entities.

# Database Schema

# 

# Database Credentials

# MYSQL\_ADDON\_HOST=bofoxytoicrjg3ht0q2t-mysql.services.clever-cloud.com

# MYSQL\_ADDON\_DB=bofoxytoicrjg3ht0q2t

# MYSQL\_ADDON\_USER=ujjxirs2mutw76um

# MYSQL\_ADDON\_PORT=3306

# MYSQL\_ADDON\_PASSWORD=6MTQ28uIk1EnJUdYQ7QH

# MYSQL\_ADDON\_URI=mysql://ujjxirs2mutw76um:6MTQ28uIk1EnJUdYQ7QH@bofoxytoicrjg3ht0q2t-mysql.services.clever-cloud.com:3306/bofoxytoicrjg3ht0q2t

# Database Script

USE bofoxytoicrjg3ht0q2t;

CREATE TABLE `users` (

`user\_id` varchar(255) NOT NULL,

`full\_name` varchar(250) DEFAULT NULL,

`email` varchar(250) DEFAULT NULL,

`phone` varchar(255) DEFAULT NULL,

`gender` varchar(255) DEFAULT NULL,

`country` varchar(255) DEFAULT NULL,

`age` int(11) DEFAULT NULL,

`user\_type` varchar(255) DEFAULT NULL,

`user\_photo` varchar(250) DEFAULT NULL,

PRIMARY KEY (`user\_id`)

);

CREATE TABLE `leagues` (

`league\_id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(150) DEFAULT NULL,

`logo` varchar(255) DEFAULT NULL,

`no\_of\_teams` int(11) DEFAULT NULL,

`user\_id` varchar(255) DEFAULT NULL,

PRIMARY KEY (`league\_id`),

KEY `FK\_LeagueManager` (`user\_id`),

CONSTRAINT `FK\_LeagueManager` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `teams` (

`team\_id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(255) DEFAULT NULL,

`logo` varchar(255) DEFAULT NULL,

`team\_label` varchar(50) DEFAULT NULL,

`user\_id` varchar(255) DEFAULT NULL,

PRIMARY KEY (`team\_id`),

UNIQUE KEY `user\_id\_UNIQUE` (`user\_id`),

CONSTRAINT `FK\_TeamManager` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `players` (

`player\_id` int(11) NOT NULL AUTO\_INCREMENT,

`full\_name` varchar(250) DEFAULT NULL,

`player\_photo` varchar(250) DEFAULT NULL,

`position` varchar(100) DEFAULT NULL,

`strength` varchar(100) DEFAULT NULL,

PRIMARY KEY (`player\_id`)

);

CREATE TABLE `schedules` (

`schedule\_id` int(11) NOT NULL AUTO\_INCREMENT,

`location` varchar(150) DEFAULT NULL,

`date\_of\_match` varchar(50) DEFAULT NULL,

`time\_of\_match` varchar(50) DEFAULT NULL,

`team1\_id` int(11) DEFAULT NULL,

`team2\_id` int(11) DEFAULT NULL,

`league\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`schedule\_id`),

KEY `FK\_ScheduleMatch` (`league\_id`),

KEY `FK\_Team1ScheduleMatch` (`team1\_id`),

KEY `FK\_Team2ScheduleMatch` (`team2\_id`),

CONSTRAINT `FK\_ScheduleMatch` FOREIGN KEY (`league\_id`) REFERENCES `leagues` (`league\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_Team1ScheduleMatch` FOREIGN KEY (`team1\_id`) REFERENCES `teams` (`team\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_Team2ScheduleMatch` FOREIGN KEY (`team2\_id`) REFERENCES `teams` (`team\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `matches` (

`match\_id` int(11) NOT NULL AUTO\_INCREMENT,

`result` varchar(50) DEFAULT NULL,

`schedule\_id` int(11) DEFAULT NULL,

PRIMARY KEY (`match\_id`),

KEY `FK\_MatchSchedule` (`schedule\_id`),

CONSTRAINT `FK\_MatchSchedule` FOREIGN KEY (`schedule\_id`) REFERENCES `schedules` (`schedule\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `players\_in\_team` (

`player\_id` int(11) NOT NULL,

`team\_id` int(11) NOT NULL DEFAULT '0',

PRIMARY KEY (`player\_id`,`team\_id`),

KEY `FK\_TeamPlayer` (`team\_id`),

CONSTRAINT `FK\_PlayerTeam` FOREIGN KEY (`player\_id`) REFERENCES `players` (`player\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_TeamPlayer` FOREIGN KEY (`team\_id`) REFERENCES `teams` (`team\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `league\_team` (

`league\_id` int(11) NOT NULL,

`team\_id` int(11) NOT NULL,

PRIMARY KEY (`league\_id`,`team\_id`),

KEY `FK\_TeamLeague` (`team\_id`),

CONSTRAINT `FK\_LeagueTeam` FOREIGN KEY (`league\_id`) REFERENCES `leagues` (`league\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_TeamLeague` FOREIGN KEY (`team\_id`) REFERENCES `teams` (`team\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE `scores` (

`score\_id` int(11) NOT NULL AUTO\_INCREMENT,

`match\_id` int(11) DEFAULT NULL,

`team\_id` int(11) DEFAULT NULL,

`goal` int(11) DEFAULT NULL,

`shots` int(11) DEFAULT NULL,

`shots\_on\_target` int(11) DEFAULT NULL,

`possession` int(11) DEFAULT NULL,

`passes` int(11) DEFAULT NULL,

`pass\_accuracy` int(11) DEFAULT NULL,

`fouls` int(11) DEFAULT NULL,

`yellow\_cards` int(11) DEFAULT NULL,

`red\_cards` int(11) DEFAULT NULL,

`offsides` int(11) DEFAULT NULL,

`corners` int(11) DEFAULT NULL,

PRIMARY KEY (`score\_id`),

KEY `FK\_MatchScore` (`match\_id`),

KEY `FK\_TeamScore` (`team\_id`),

CONSTRAINT `FK\_MatchScore` FOREIGN KEY (`match\_id`) REFERENCES `matches` (`match\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `FK\_TeamScore` FOREIGN KEY (`team\_id`) REFERENCES `teams` (`team\_id`) ON DELETE CASCADE ON UPDATE CASCADE

);

Insert into `users` (`full\_name`, `email`, `phone`, `gender`, `country`,

`age`, `user\_type`, `user\_photo`) values

('Robert', 'robert@gmail.com', 514857, 0, 'CANADA', 23, 0, 'No photo'),

('John', 'john@gmail.com', 51485790, 0, 'INDIA', 23, 0, 'No photo');

Insert into `teams` ( `name`, `logo`, `team\_label`,`user\_id`) values

('mumbaiindien', 'No photo', 'MI',1),

('kolkataknightrider', 'No photo', 'kk',2);

Insert into `leagues` (`name`, `logo`, `no\_of\_teams`, `user\_id`) values

('MTL', 'No photo', 8, 1),

('ABD', 'No photo', 10, 1);

Insert into `players` (`full\_name`, `player\_photo`, `position`, `strength`) values

('parag', 'No photo', 'forward', '70'),

('yash', 'No photo', 'midfielder', '70');

Insert into `players\_in\_team` (`player\_id`, `team\_id`) values

(1, 1),

(2, 1);

Insert into `league\_team` (`league\_id`, `team\_id`) values

(1, 1),

(1, 2);

INSERT into schedules(Location,date\_of\_match,Time\_of\_match,Team1\_id,Team2\_id,League\_id)

VALUES ("Banglore" , "2020-02-25" , "5:00 PM" , 1 , 2 , 1);

INSERT into schedules(Location,date\_of\_match,Time\_of\_match,Team1\_id,Team2\_id,League\_id)

VALUES ("Indore", "2019-10-10" , "5:00 PM" , 1 , 2 , 2);

Insert into `scores` ( `match\_id`, `team\_id`, `goal`, `shots`, `shots\_on\_target`,

`possession`, `passes`, `pass\_accuracy`, `fouls`, `yellow\_cards`, `red\_cards`, `offsides`,

`corners`) values

(1, 1, 3, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1),

(1, 2, 2, 2, 3, 2, 1, 2, 2, 3, 2, 2, 1);

Insert into `matches` (`result`, `schedule\_id`) values ('winner', 1), ('winner', 2);

1. **Web Services’ Calls Details**

* Add Team in League

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/AddTeamInLeague&1&1> |
| Call | Add Team in League |
| Parameters | leagueid = 1  teamid = 1 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “Team successfully added in league.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong! team is not added in League.”  } |

* Create League

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/createLeague&APL&NoPhoto&11&1> |
| Call | Create League |
| Parameters | name = APL  logo = NoPhoto  noOfTeams = 11  userId = 1 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “League successfully created.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong! League not created.”  } |

* League List by User

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/LeagueListByUser&1> |
| Call | League List by User |
| Parameters | userid = 1 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “LeagueList”: [  {  “League Id”: 1,  “Name” : “MTL”,  “logo” : “No photo”,  “noOfTeams” : 8,  “”UserID” : 1  },  {  “League Id”: 2,  “Name” : “ABD”,  “logo” : “No photo”,  “noOfTeams” : 10,  “”UserID” : 1  },  {  “League Id”: 5,  “Name” : “IFL”,  “logo” : “NoPhoto”,  “noOfTeams” : 24,  “”UserID” : 2  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “No Records Found.”  } |

* Remove Team from League

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/RemoveTeamFromLeague&8&3> |
| Call | Remove Team from League |
| Parameters | leagueid = 8  teamid = 3 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547143,  “message” : “Team successfully deleted from league.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong! Team is not deleted from league.”  } |

* Update League

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/updateLeague&6&NPL&NoPhoto&12&1> |
| Call | Update League |
| Parameters | leagueid = 7  name = CPL  logo = NoPhoto  noOfTeams = 10  userId = 3 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “League successfully updated.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong! League not updated.”  } |

* Played Matches by LeagueId

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/PlayedMatchesByLeagueId&2> |
| Call | Played Matches by LeagueId |
| Parameters | leagueid = 2 |
| Response | **Example 1:**  {  “LeagueList” : [  {  “Status” : 200,  “TimeStamp” : 1581547017  “League Id” : 2,  “Match Id”: 2,  “Schedule Id” : 2,  “Team1 Id”: 1,  “Team2 Id”: 2,  “Date of Match” : “2019-05-27”,  “Time of Match”: “15:00 PM”,  “Location” : “USA”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “No Records Found.”  } |

.

* Match Score

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/matchScore&1&2> |
| Call | Match Score |
| Parameters | matchid = 1  teamid = 2 |
| Response | **Example 1:**  {  “MatchScore” : [  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Score Id” : 2,  “Match Id”: 1,  “Team Id” : 2,  “goal”: 5,  “shots”:8,  “shotsOnTarget” : 9,  “possession”: 6,  “passes” : 5,  “passAccuracy”: 4,  “fouls” : 7,  “yellowCards”: 8,  “redCards”: 9,  “offsides” : 6,  “corners” : 3  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “No Records Found.”  } |

* List of Country

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/ListOfCountries> |
| Call | List of Country |
| Parameters | No Parameters |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Countries”: [  “CANADA”,  “INDIA”  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong!!”  } |

* Create Match Schedule

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/CreateSchedule&Toronto&2020-05-27&15:00PM&3&5&8> |
| Call | Create Match Schedule |
| Parameters | Location= Toronto  Date\_Of\_Match = 2020-05-27  Time\_Of\_Match = 15 : 00 PM  Team1\_id = 3  Team2\_id = 5  League\_id = 8 |
| Response | **Example 1:**  **{**  “Status”: 200,  “TimeStamp”: 1581543891,  “Message”: “Schedule for The Match Is Fixed!!!”  **}**  **Example 2:**  {  “status”: “Wrong”,  “timestamp”:1571140913,  “message”: “Something Went Wrong!!”  } |

* Reschedule Match

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/ReSchedule&USA&2020-05-27&15:00PM&3&5&8&7> |
| Call | Reschedule Match |
| Parameters | Location= USA  Date\_Of\_Match = 2020-05-27  Time\_Of\_Match = 15: 00 PM  Team1\_id = 3  Team2\_id = 5  League\_id = 8  Schedule\_id = 7 |
| Response | **Example 1:**  **{**  “Status”: 200,  “TimeStamp”: 1581544688,  “Message”: “Match is Reschedule!!!”  **}**  **Example 2:**  {  “status”: “Wrong”,  “timestamp”:1571140913,  “message”: “Something Went Wrong!!”  } |

* Create Team

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/CreateTeam&WestIsland&nophoto&WI&2> |
| Call | Create new team |
| Parameters | name = WestIsland  logo = nophoto  team\_label = WI  user\_id = 2 |
| Response | **Example 1:**  **{**  "Status": 200,  "TimeStamp": 1581547344,  "Message": "Team created"  **}**  **Example 2:**  {  “status”: “Wrong”,  “timestamp”:1571140913,  “message”: “Something Went Wrong!!”  } |

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/viewPlayerListFromTeam&1> |
| Call | Displays player list through team id |
| Parameters | Team\_id = 1 |
| Response | **Example 1:**  **{**  "Status": 200,  "TimeStamp": 1581544688,  "Player List: [  {  "Full name": “parag”,  "Player Photo": "No photo",  “Strength” : 70  },  {  "Full name": “yash”,  "Player Photo": "No photo",  “Strength” : 70  }  ]  }  **Example 2:**  {  “status”: “Wrong”,  “timestamp”:1571140913,  “message” : "Records of players not found”.  } |

* View Player List from Team
* View Team List

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/viewTeamList> |
| Call | View Team List |
| Parameters | No Parameter |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Team List” : [  {  “Team Id” : 1,  “Team name ”: “mumbaiindien”,  “Logo” : “No photo”,  “Team Label”: “MI”,  “User Id”: 1  },  {  “Team Id” : 2,  “Team name ”: “kolkataknightrider”,  “Logo” : “No photo”,  “Team Label”: “kk”,  “User Id”: 2  },  {  “Team Id” : 3,  “Team name ”: “WestIsland”,  “Logo” : “No photo”,  “Team Label”: “WI”,  “User Id”: 2  },  {  “Team Id” : 4,  “Team name ”: “WestIsland”,  “Logo” : “No photo”,  “Team Label”: “WI”,  “User Id”: 2  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Record of Teams not found.”  } |

* View Team List from LeagueID

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/viewTeamListFromLeagueId&2> |
| Call | View Team List from LeagueId |
| Parameters | leagueid = 1 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Team List” : [  {  “Team Name”: “mumbaiindien”,  “Logo” : “No photo”  },  {  “Team Name”: “kolkataknightrider”,  “Logo” : “No photo”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Records of Teams using league\_id not found.”  } |

* Register User

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/registerUser&John&john1@gmail.com&5147788987&john123&0&India&22&1&noPhoto> |
| Call | Register User |
| Parameters | full\_name = John  email = john1@gmail.com  phone = 5147788987  gender = 0  country = India  age = 22  usertype = 1  userphoto = NoPhoto |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “User successfully registered.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “User not registered.”  } |

* Upcoming Matches-Guest Dashboard

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/upcomingMatches_guestDashboard> |
| Call | Upcoming Matches |
| Parameters | No Parameters |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “UpcomingMatchList” : [  {  “name”: “mumbaindien”,  “date\_of\_match”: “2020-02-25”,  “logo” : “No Photo”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “No Records Found.”  } |

* Played Matches-Guest Dashboard

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/playedMatches_guestDashboard> |
| Call | Played Matches |
| Parameters | No Parameters |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “PlayedMatchList” : [  {  “name”: “mumbaiindien”,  “date\_of\_match”: “2019-10-10”,  “logo” : “No photo”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “No Records Found.”  } |

* Add Player in Team

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/AddPlayerInTeam&Jiny&nophoto&goalkeeper&79&2> |
| Call | Add Player in Team |
| Parameters | full\_name = Jiny  player\_photo = NoPhoto  position = goalkeeper  strength = 79  user\_id = 2 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “Player added to the team.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Player not added to the team.”  } |

* Modify Player Details

|  |  |
| --- | --- |
| URL | [https://soccerallianceapp.appspot.com/rest/api/ModifyPlayerDetails&8 &shibit&Nophoto&fielder&55](https://soccerallianceapp.appspot.com/rest/api/ModifyPlayerDetails&8&shibit&Nophoto&fielder&55) |
| Call | Modify Player Details |
| Parameters | player\_id = 8  full\_name = shibit  player\_photo = NoPhoto  position = fielder  strength = 55 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “Player detail modified.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Player detail not modified.”  } |

* League Details by League\_ID

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/leagueDetailsByleagueID&2> |
| Call | League Details by League\_id |
| Parameters | League\_id = 2 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Leagues” : [  {  “league\_id” : 2,  “name” : “ABD”,  “logo” : “NoPhoto”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “League not found.”  } |

* League List by Country

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/ListOfLeaguesByCountry&CANADA> |
| Call | League List by Country |
| Parameters | Country = CANADA |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “Leagues” : [  {  “id” : 1,  “name” : “MTL”  },  {  “id” : 2,  “name” : “ABD”  }  ]  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “fail to get country…”,  “Leagues:” : []  } |

* Remove Player from Team

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/removePlayerFromTeam&44&17> |
| Call | Remove Player from Team |
| Parameters | player\_id = 44  team\_id = 17 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “result” : “player removed from team.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “result” : “System won’t be able to remove player!!!”  } |

* Update User Profile

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/UpdateUserProfile> |
| Call | Update User Profile |
| Parameters | user\_id = VQRUZDSIgTWuEPfbffedpXYD7w73  full\_name = jay  email = j2jayyy@gmail.com  phone = 5146587474  gender = male  country = india  age = 24  user\_type = league manager |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “Your profile has been updated.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “result” : “There is some problem.”  } |

* Cancel Match

|  |  |
| --- | --- |
| URL | <https://soccerallianceapp.appspot.com/rest/api/CancelMatch&15> |
| Call | Cancel Match |
| Parameters | match\_id = 15 |
| Response | **Example 1:**  {  “Status” : 200,  “TimeStamp” : 1581547017,  “message” : “Match has been cancelled due to bad weather.”  }  **Example 2:**  {  “Status” : “Error”,  “TimeStamp” : 1581547017,  “message” : “Something went wrong.”  } |

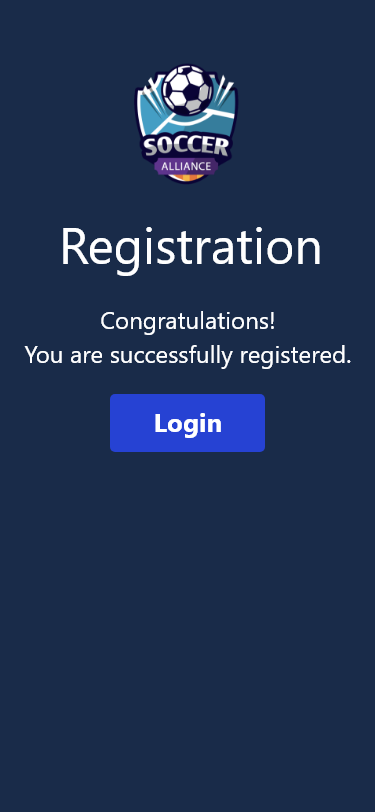
# Prototype

# Splash Screen & login & forgot password

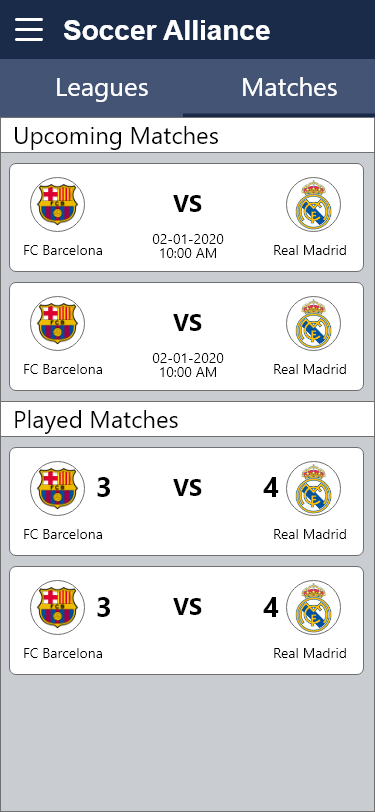
# F:\MAD\Semester-4\Soccer management app\Prototype\Prototype pics\Forgot password.png

# Registration Details

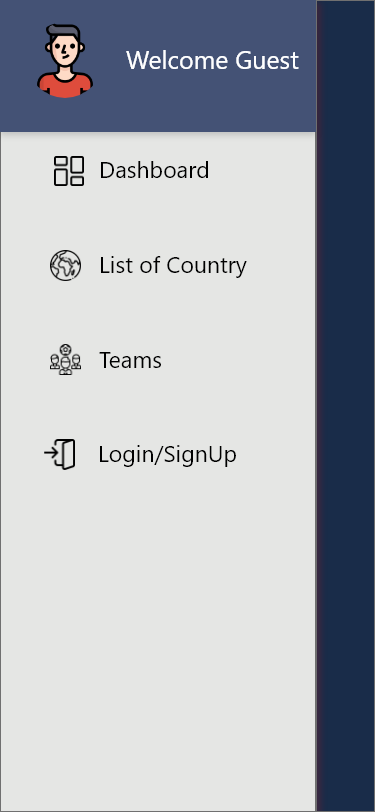
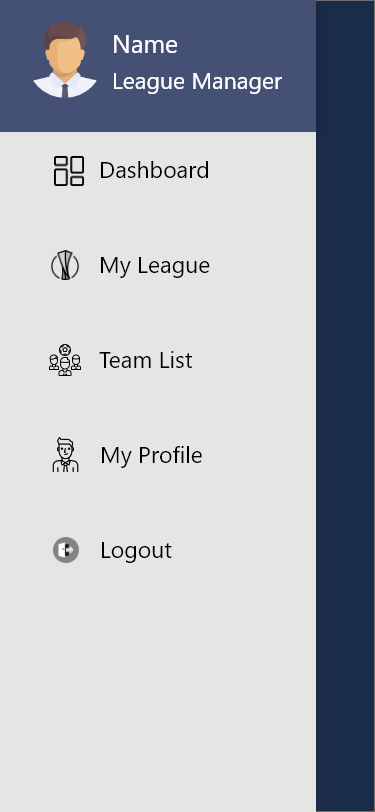
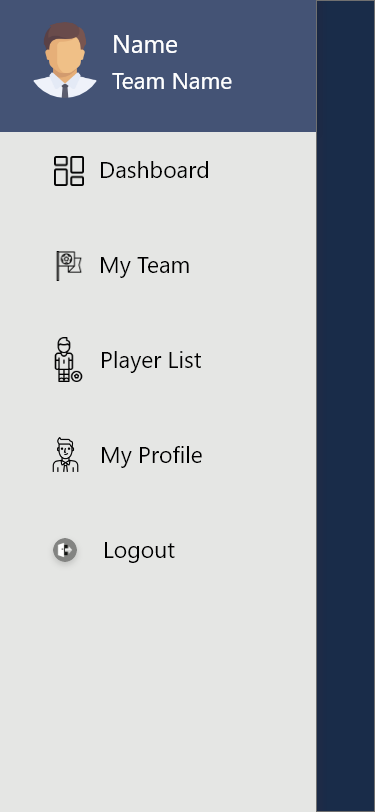
# F:\MAD\Semester-4\Soccer management app\Prototype\Prototype pics\Registration – Email-contact.png Registration – Personal Details F:\MAD\Semester-4\Soccer management app\Prototype\Prototype pics\Registration - Password.png

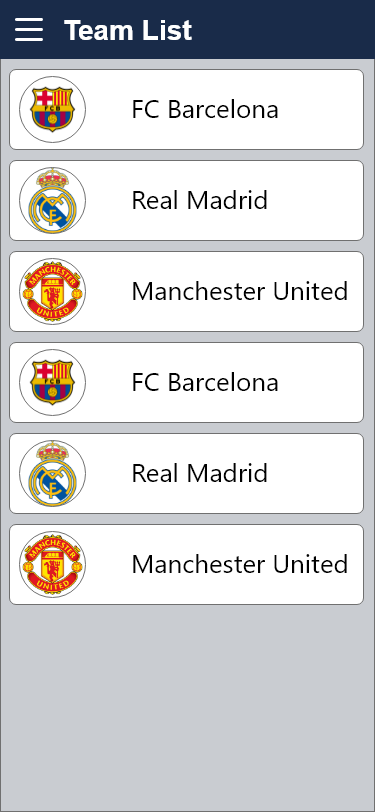


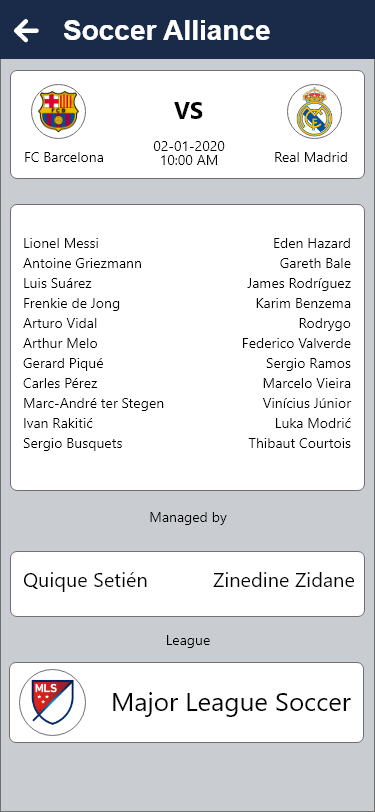
* Dashboard

* All Drawer Menu

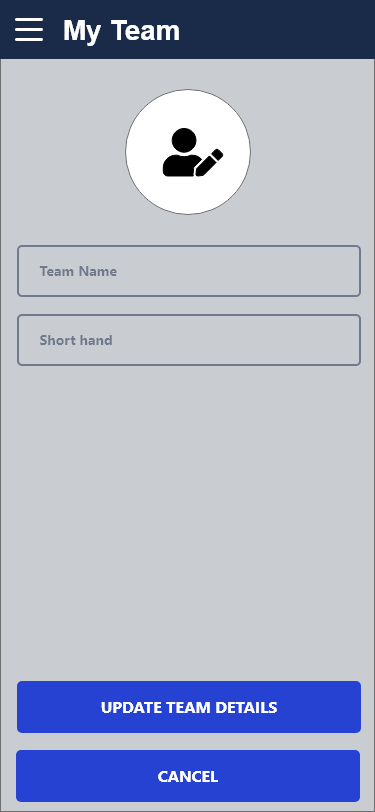
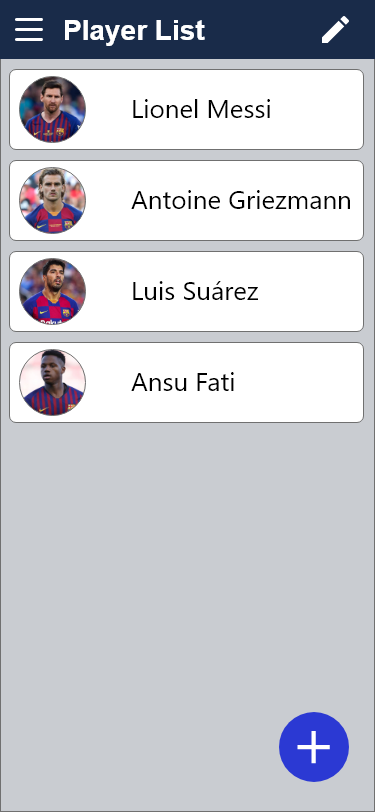
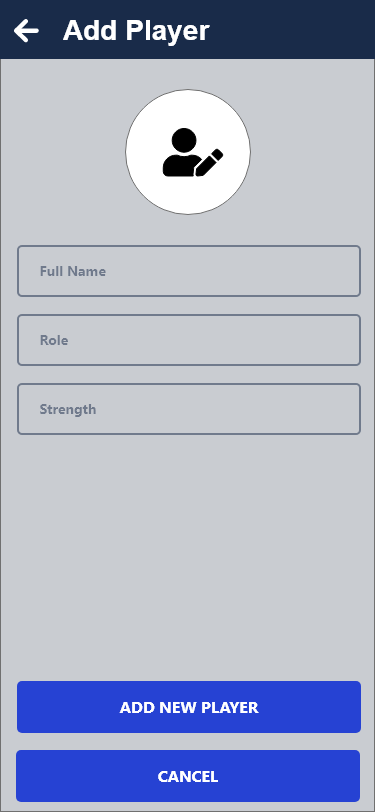
  

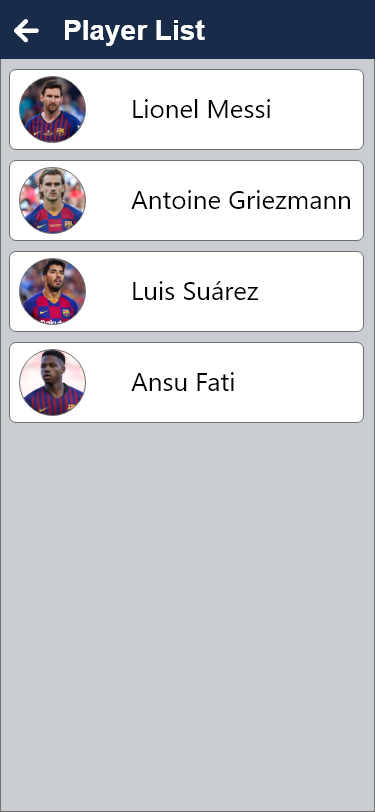
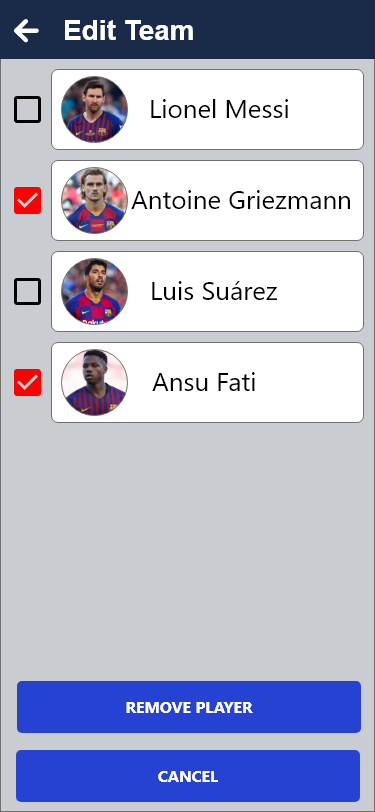
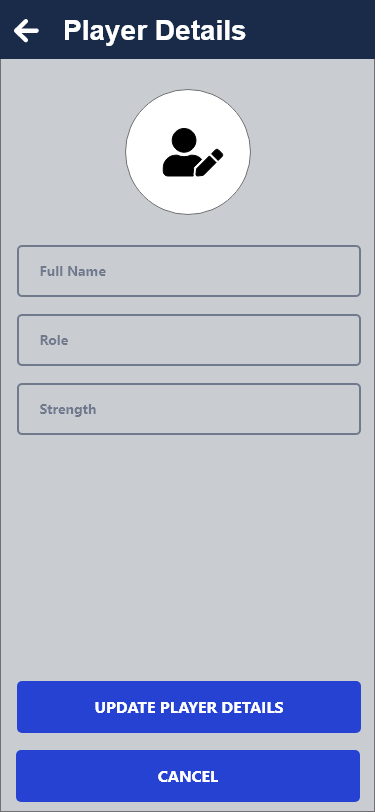
  

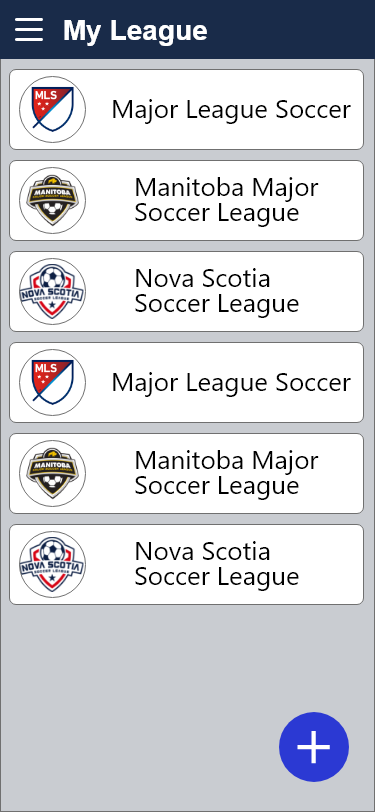
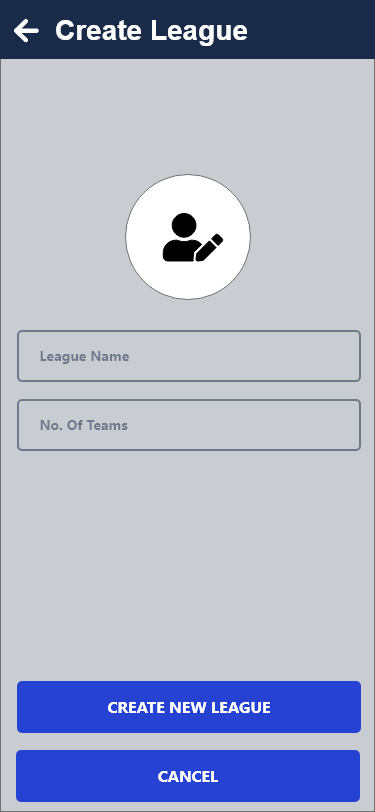
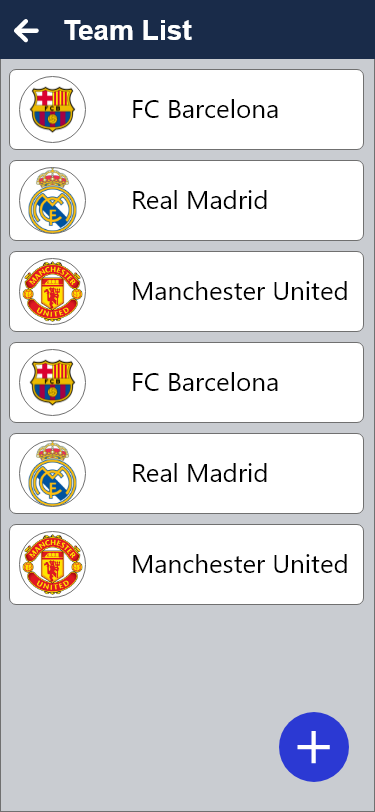
* My Profile

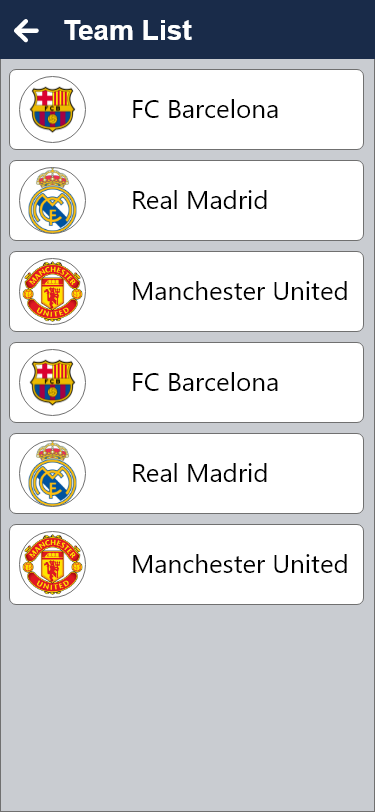


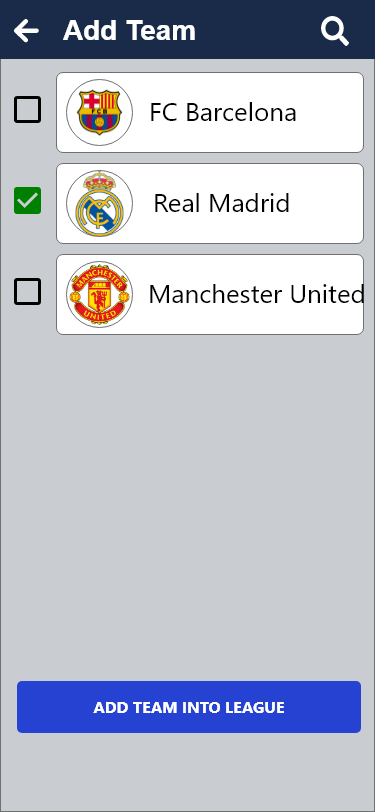
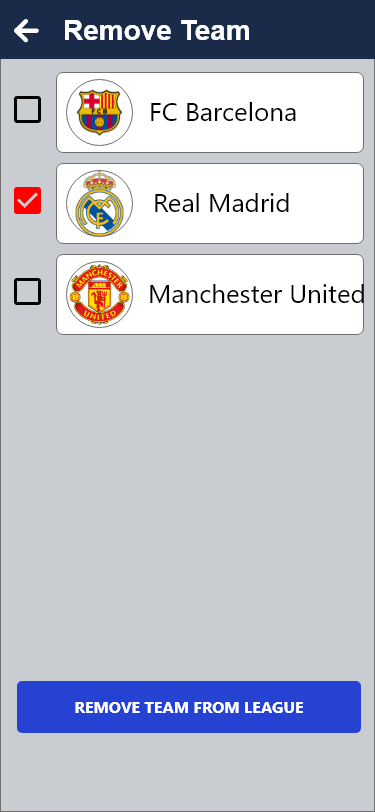
  

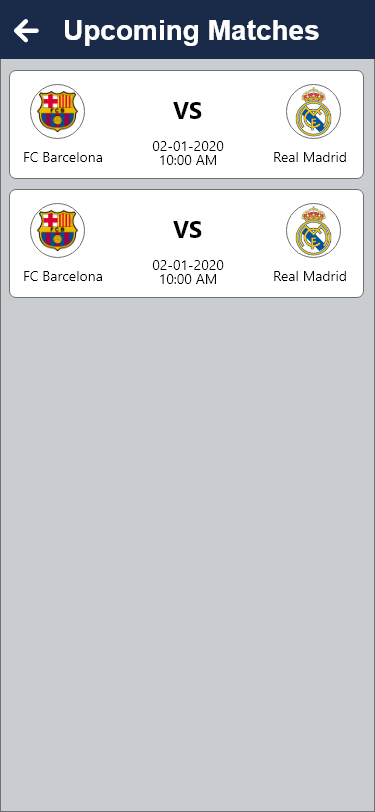
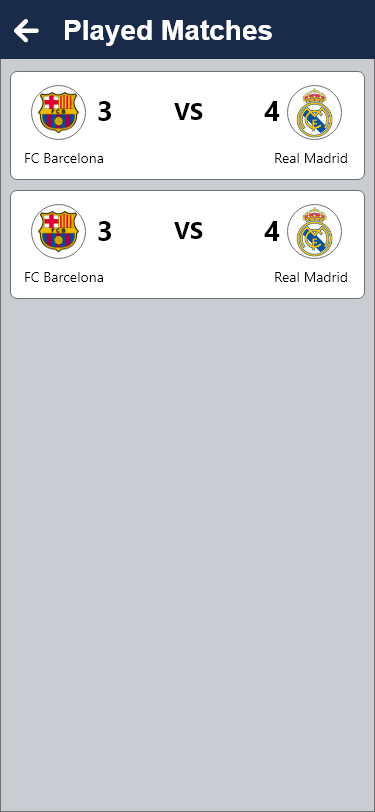
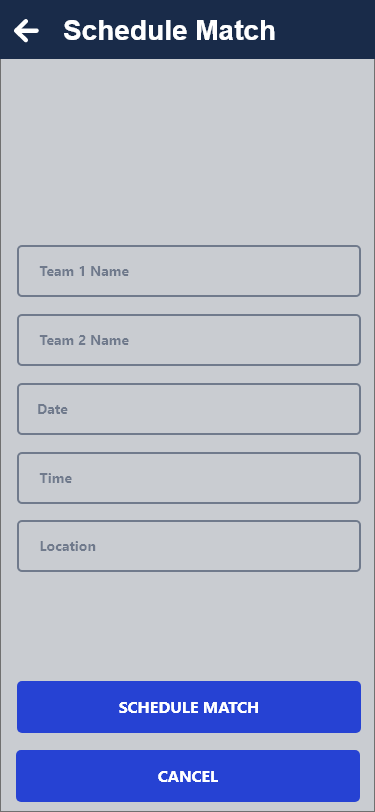
  

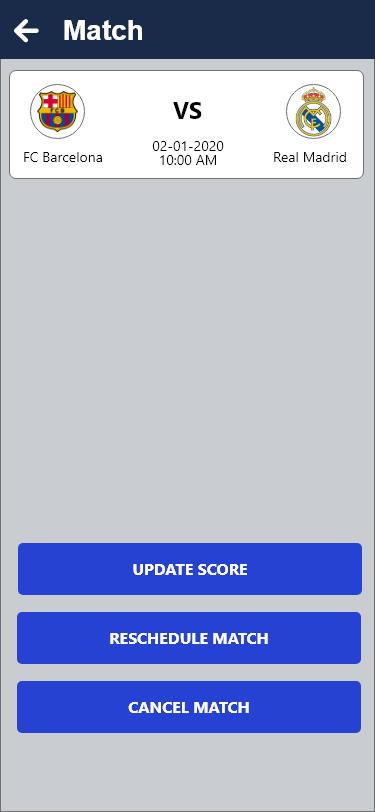
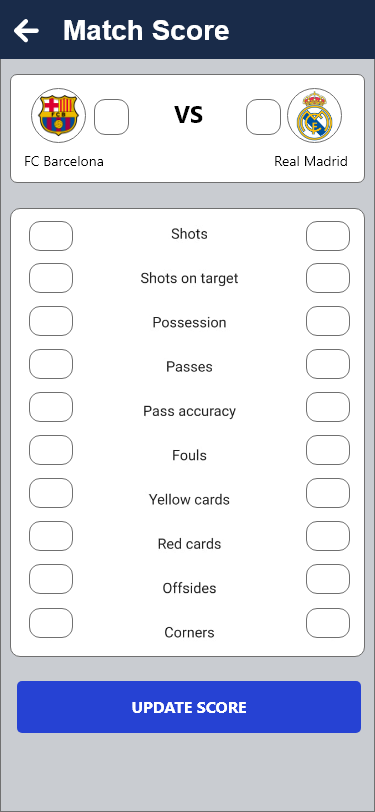
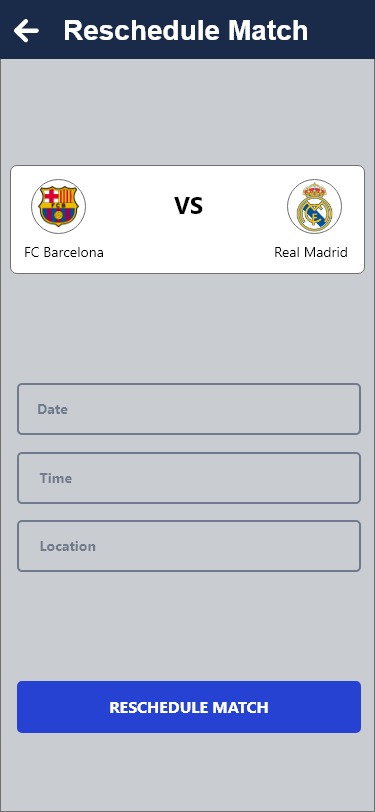
* League

# References

# <https://firebase.google.com/docs/android/setup?authuser=0>

# <https://developer.android.com/docs>

# <https://stackoverflow.com/questions/55449935/how-to-check-the-internet-connection-continuously-while-app-is-running>

# <https://www.tutlane.com/tutorial/android/android-internet-connection-status-with-examples>

# <https://cloud.google.com/apis/design/documentation>

# <https://cloud.google.com/appengine/docs/standard/java>

# <https://square.github.io/retrofit/>

# <https://github.com/bumptech/glide>

# <https://github.com/material-components/material-components-android>

# <https://github.com/wasabeef/awesome-android-ui>