A close up of a sign

Description automatically generated

**Report File**

**Guided By:**

Prof. [*Name of Professor*]

Prof. [*Name of Professor*]

**Prepared By:**

*Amandeep Singh 1896409*

*Varinder Singh Sandhu 1896305*

*Sandeep Singh Sidhu 1894980*

*Harbir Singh 1895897*

*Amandeep Kaur 1794977*

*Parminder Kaur 1896181*

Table of Contents

[1. Introduction 3](#_Toc41239982)

[1.1 Purpose 3](#_Toc41239983)

[1.2 Scope 3](#_Toc41239984)

[2. General Description 4](#_Toc41239985)

[2.1 Product Functions 4](#_Toc41239986)

[2.2 User Characteristics 4](#_Toc41239987)

[3. Specific Requirements 5](#_Toc41239988)

[3.1 Functional Requirements 5](#_Toc41239989)

[3.2 Non-Functional Requirements 7](#_Toc41239990)

[4. Analysis Models 8](#_Toc41239991)

[4.1 Use Case Diagram 8](#_Toc41239992)

[4.2 Use Case Scenarios 9](#_Toc41239993)

[4.3 Class Diagram 41](#_Toc41239994)

[5. Entity relationShip diagram](#_Toc41239997) 42

[5.1 Database Schema 42](#_Toc41239998)

[5.2 Logical schema of firebase database 43](#_Toc41239999)

**5.3** Database Script**..............................................................................................................44**

**6.Screen Designs..**........................................................................................................................51

# **Introduction**

This document describes the software requirements of the mobile application refer to as a Soccer Management App. It gives a general description of the project, the functional and non-functional requirements, interface design, performance factor, and database requirement, use case diagram and GUI Design.

## **Purpose**

This Software Requirement Specification (SRS) specifies the requirements of the Cegep Gim Soccer League application, which will be used by Guest Users, Team Managers and League Managers. In addition, the main purpose of this application is to maintain the records of all teams, team managements and score records regarding winning or losing and total score of teams.

## **Scope**

Cegep Gim Soccer League is the mobile application, which is useful for all team managers, league managers, team players and guest users as described here. Guest users can see all the team details without login, but guest user cannot modify any details. Team manager can manage only one team assigned by league manager. While League managers have right to manage all team’s details, team manager’s details and score of the game.

# **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.

## **Product Functions**

* Login
* Sign up
* Forget Password
* Guest Users
* Manage Team Managers
* Manage Players
* Manage Seasons (Leagues)
* Manage Schedules
* Manage Teams
* Modify Team Info
* Manage Score Board.

## **User Characteristics**

Cegep Gim Soccer League is available for all cegep gim college’s students who uses android mobile device.

The end-users of Cegep Gim Soccer League are student of Cegep Gim college who is taking participate in league, guests and all managers.

All guest users can visit the app and see all details in the app like team info, player info, score board and statistics of all players or teams but can’t modify any details.

Team Managers can manage one and only one dedicated team. Team Manager can add or update player’s info as well as team’s information like name, color, and contact info of manager. Team manager can prepare their squad as they want. League manager have same rights as team manager, but they have access to all teams and players. League manager can create and delete teams. League can create seasons and schedules for season. They have rights to modify / enter results of matches.

# **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 **Cegep Gim Soccer League** Application.

|  |  |  |  |
| --- | --- | --- | --- |
| Functional Requirements | | | |
| Sr. No | **Name** | **Description** | **Must/Could** |
| FR-1 | Login | To use this app, all users must login using user’s Email id and password. It allows user to access al premium feature of application. Team Manager and League manager has over all access of app. | MUST |
| FR-2 | Sign up | This app must allow user to register them. It requires First Name, Last Name, Email id, Password. | MUST |
| FR-3 | Forget Password | This app must allow user to reset password incase if user forget. For that user must have to enter registered email and firebase will send a password reset link on registered email via email where user can enter new password. | MUST |
| FR-4 | Guest Users | This app must allow guest user to browse scores, leagues, teams, players and team details but not allowed to modify that details. | MUST |
| FR-5 | Manage Team Managers | This app must allow only league manager to add or remove team managers. | MUST |
| FR-6 | Manage Players | This app must allow only team manager to add or remove players with details like player’s first name, player’s last name, player’s age, player’s profile image, and player’s position. | MUST |
| FR-7 | Manage Seasons (Leagues) | This app must allow only league manager to add or remove seasons (leagues). | MUST |
| FR-8 | Manage Schedules | This app must allow only league manager to add or update schedules of seasons. For that league’s manager must select team1 and team 2, league, date, time, and place. | MUST |
| FR-9 | Manage Teams | This app must allow only league manager to add or remove teams. For that league manager must enter team name, team icon, and select team manager for that particular team. | MUST |
| FR-10 | Modify Team Info | Team manager can edit team info and team squad. | COULD |
| FR-11 | Manage Score Board | This app must allow only league manager to add or update score board. | MUST |

## **Non-Functional Requirements**

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

|  |  |  |
| --- | --- | --- |
| Non-Functional Requirements | | |
| Sr. No | **Name** | **Description** |
| NFR-1 | Performance Requirements | The application shall be fast and robust when loading. The application should not take more than 5 seconds to respond. |
| NFR-2 | Space Requirements | The device shall have enough memory space to load the app. |
| NFR-3 | Usability Requirements | The application shall be easy to use and understand. The application shall have a user-friendly interface. GUI shall be simple and clear. |
| NFR-4 | Reliability requirements | The application shall not produce an incorrect output in any manner. |
| NFR-5 | Implementation Requirements | The application shall be implemented using Android Studio 3.6 and Firebase services. |
| NFR-6 | Security Requirements | The application shall not allow unauthorized login attempts more than 3 times. |
| NFR-7 | Maintainability Requirements | The application shall allow making changes quickly and costing effectively. New changes will improve the efficiency of the application. |
| NFR-8 | Interoperability Requirements | The application shall allow access to the different pages of the application without altering its efficiency and consistency. |
| NFR-9 | Privacy Requirements | Personal Information shall be only accessed by information owner. |

# **Analysis Models**

This section of the document defines the structural diagram and flow of the **Cegep Gim Soccer League** application.

## **Use Case Diagram**

A close up of text on a white background

Description automatically generated

## **Scenarios**

* **Login**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-1 | |
| Use Case Name | Login | |
| Date created | 22/05/2020 | |
| Actor | Registered user (Team Manager and League Manager) | |
| Description | To perform any kind of task All users have to login first. Without login user cannot perform any task. | |
| Triggering event | User clicks on login button. | |
| 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 home screen according to their role. |
| Alternative flow of events | Actor | System |
| 1. Enter email and password. 2. Click on Login Button. 3. Repeat step 1 and 2. | 1. Verify entered email and password in Firebase Authentication. 2. Notify the user with message, “Email or password you have entered is incorrect!” 3. Clear entered email and password field. |
| Post conditions | User have now logged in into system | |
| Exception Conditions | User does not have an account. | |

* **SignUp**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-2 | |
| Use Case Name | Sign up | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can register in application using sign up page. | |
| Triggering event | League Manager clicks on the Sign-up button from signup page. | |
| Preconditions | League Manager does not have an account. | |
| Flow of events | Actor | System |
| 1. Enter First name 2. Enter Last name 3. Enter Email Id 4. Enter Password. 5. Click on Sign up Button. | 1. Check email id is not registered in firebase authentication. 2. Notify the user with message, “Registration Successfully”. 3. Redirect User to login screen. |
| Alternative flow of events | Actor | System |
| 1. Enter First name 2. Enter Last name 3. Enter Email ID 4. Enter Password. 5. Click on Sign up Button. 6. Repeat Step 1 to 5 | 1. Check Email id is already registered or not with firebase authentication. 2. Notify the user with message “This email id is already registered! Please enter another email id.” 3. Clear entered all data in field. |
| Post conditions | League Manager now has an account. | |
| Exception  Conditions | League Manager is already registered and try to register using already registered email id. | |

* **Forget Password**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-3 | |
| Use Case Name | Forget Password | |
| Date created | 22/05/2020 | |
| Actor | League Manager or Team Manager | |
| Description | League Manager and Team Manager can reset their password. | |
| Triggering event | League Manager or Team Manager clicks on send verification link button. | |
| Preconditions | User must have an account. | |
| Flow of events | Actor | System |
| 1. Enter Registered Email Id 2. Click on Send password reset link button | * 1. Check Email id is valid and registered in firebase authentication.   2. Send Password Reset link to entered email id.   3. Notify the user with message “Password reset Link send successfully!”   4. Redirect user to Login Page. |
| Alternative flow of events | Actor | System |
| 1. Enter Registered Email Id 2. Click on Send password reset link button 3. Repeat Step 1 and 2. | * 1. Check Email id is valid and registered in firebase authentication   2. Notify the user with message “Please enter Valid Email id!”   3. Clear value of email field. |
| Post conditions | League Manager or Team manager can reset password by clicking received link in email from Firebase Authentication. | |
| Exception  Conditions | League manager and team manager do not have an account. | |

* **Update Profile**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-4 | |
| Use Case Name | Update Profile | |
| Date created | 22/05/2020 | |
| Actor | League Manager or Team Manager | |
| Description | League manager or Team manager can update their First name, Last name, Password. | |
| Triggering event | League manager or Team manager clicks on Update profile button. | |
| Preconditions | User must be already logged in the application. | |
| Flow of events | Actor | System |
| 1. Select Update profile from side menu on Home screen. 2. User modifies the information. 3. Click on update details button. | * 1. Get user’s personal information from database.   2. Show user’s Personal information.   3. Check if user has modified any information.   4. Update user’s information in database.   5. Notify user with message, “Profile updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. Select Update profile from side menu on Home screen. 2. User modifies the information. 3. Click on update details button. 4. Repeat step 1 to 3 | * 1. Get user’s personal information from database.   2. Show user’s Personal information.   3. Check if user has modified any information.   4. Notify user with message, “To update profile, the information must have been changed!” |
| Post conditions | League Manager or Team manager have changed their personal information. | |
| Exception | User is not logged into system. | |

* **Create league**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-5 | |
| Use Case Name | Create League | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can create league. | |
| Triggering event | League Manager clicks on Add League button. | |
| Preconditions | League Manager must be logged into system. | |
| Flow of events | Actor | System |
| 1. User Select Logo for League 2. User enter league name 3. Clicks on Add league button | * 1. System verifies data which is entered by user.   2. Check League is not already registered!   3. Save Logo and league name in database   4. Notify the user with message “League added successfully!” |
| Alternative flow of events | Actor | System |
| 1. User Select Logo for League 2. User enter league name 3. Clicks on Add league button 4. Repeat step 1 to 3. | * 1. System verifies data which is entered by user.   2. Check League is not already registered!   3. Notify the user with message “This league name is already registered.”   4. Clear all field for user to enter correct information. |
| Post conditions | League manager has successfully added new league. | |
| Exception | League manager is not allowed to add league without league name, and logo. | |

* **Update League**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-6 | |
| Use Case Name | Update League | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can Update league. | |
| Triggering event | When League Manager clicks on Update League button. | |
| Preconditions | League Manager must have logged in system. | |
| Flow of events | Actor | System |
| 1. User Select League and click on edit league button 2. User modifies the logo or league name. 3. Clicks on update league button | * 1. Get selected league’s information.   2. Show that league’s information.   3. Check if user has modified any information.   4. Update league’s information in database.   5. Notify the user with message, “League’s information updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User Select League and click on edit league button 2. User modifies the data logo or league name. 3. Clicks on update league button 4. Repeat step 1 to 3. | * 1. Get selected league’s information.   2. Show that league’s information.   3. Check if user has modified any information.   4. Notify user with message, “To update league, the information must have been changed!” |
| Post conditions | League manager has successfully updated league. | |
| Exception | League manager has not changed anything in information. | |

* **Add Team**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-7 | |
| Use Case Name | Add Team | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can Add new team and team manager. | |
| Triggering event | League Manager clicks on add team button. | |
| Preconditions | League Manager must have logged in system. | |
| Flow of events | Actor | System |
| 1. User Select Logo For team 2. User enter team name 3. User enter team manager’s first name 4. User enter team manager’s last name 5. User enter team manager’s contact number 6. User enter team manager’s Email id 7. User enter team manager’s password 8. User enter team manager’s confirm password 9. Clicks on Add team button | * 1. System verifies data which is entered by user.   2. Check Team or Team manager is not already registered!   3. Save Logo and team name and team manager’s info in database   4. Notify user with message “Team and Team manager added successfully!” |
| Alternative flow of events | Actor | System |
| 1. User Select Logo For team 2. User enter team name 3. User enter team manager’s first name 4. User enter team manager’s last name 5. User enter team manager’s contact no 6. User enter team manager’s Email id 7. User enter team manager’s password 8. User enter team manager’s confirm password 9. Clicks on Add team button 10. Repeat step 1 to 9. | * 1. System verifies data which is entered by user.   2. Check Team or Team manager is not already registered!   3. Send Message to user “Team and Team manager’s info is already registered!” |
| Post conditions | League manager has successfully added new team and its manager. | |
| Exception  Conditions | League manager is not allowed to add new team or team manager’s information without team information like team name, and team logo. | |

* **Update Team Information**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-8 | |
| Use Case Name | Update Team info | |
| Date created | 22/05/2020 | |
| Actor | Team Manager | |
| Description | Team manager can update team’s information. | |
| Triggering event | When Team Manager clicks on Update team button. | |
| Preconditions | Team Manager must have logged in system. | |
| Flow of events | Actor | System |
| 1. User Select team and click on edit button 2. User updates information. 3. Clicks on update team button | * 1. Get selected team’s information.   2. Show that team’s information.   3. Check if user has modified any information.   4. Update team’s information in database.   5. Notify user with message, “Team’s information updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User Select team and click on edit button 2. User updates information. 3. Clicks on update team button 4. Repeat step 1 to 3. | * 1. Get selected team’s information.   2. Show that team’s information.   3. Check if user has modified any information.   4. Notify user with message, “To update team, the information must have been changed!” |
| Post conditions | Team manager has successfully updated team’s information. | |
| Exception  Conditions | Team manager is not allowed to update team’s information without different name or logo. | |

* **Delete Team**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-9 | |
| Use Case Name | Delete Team | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can delete team. | |
| Triggering event | League Manager clicks on delete button on team’s page. | |
| Preconditions | League Manager must be logged into system. | |
| Flow of events | Actor | System |
| 1. User chooses one team from list of all teams. 2. User clicks on delete button on team information page. | * 1. Get selected team’s information.   2. Show that team’s information.   3. Check team is not in ongoing league.   4. Delete team’s information from database.   5. Notify user with message, “Team deleted successfully!” |
| Alternative flow of events | Actor | System |
| 1. User chooses one team from list of all teams. 2. User clicks on delete button on team information page. 3. Repeat Step 1 and 2. | * 1. Get selected team’s information.   2. Show that team’s information.   3. Check team is not in ongoing league.   4. Notify user with message, “You cannot delete team from ongoing league.” |
| Post conditions | League manager has successfully deleted team. | |
| Exception Conditions | League manager is not allowed to delete any team from ongoing league. | |

* **Add Player**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-10 | |
| Use Case Name | Add Player | |
| Date created | 22/05/2020 | |
| Actor | Team Manager | |
| Description | Team manager can add new player in team. | |
| Triggering event | Team manager clicks on add player in team button. | |
| Preconditions | Team manager must be logged into system. | |
| Flow of events | Actor | System |
| 1. User Select profile image for player on add player screen. 2. User enter player’s first name 3. User enter player’s last name 4. User enter player’s age 5. User enter player’s position 6. Clicks on Add player in team button | * 1. System verifies data which is entered by user.   2. Check player is not already added in team!   3. Save player’s info in database   4. Notify user with message “Player added successfully!” |
| Alternative flow of events | Actor | System |
| 1. User Select profile image for player on add player screen. 2. User enter player’s first name 3. User enter player’s last name 4. User enter player’s age 5. User enter player’s position 6. Clicks on Add player in team button 7. Repeat step 1 to 6. | * 1. System verifies data which is entered by user.   2. Check player is not already added in team!   3. Notify user with message “This Player is already in team!” |
| Post conditions | Team manager has successfully added new player in team. | |
| Exception  Conditions | Team manager is not allowed to add new player without all player’s details like player’s image, first name, last name, age and position | |

**Update Team Information**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-11 | |
| Use Case Name | Update Player’s info | |
| Date created | 22/05/2020 | |
| Actor | Team Manager | |
| Description | Team manager can update player’s info. | |
| Triggering event | Team manager clicks on Update player’s details button. | |
| Preconditions | Team manager must be logged into system | |
| Flow of events | Actor | System |
| 1. User select player from list of all players in team. 2. User update player’s information 3. Clicks on update player’s details button | * 1. Get selected player’s information.   2. Show that player’s information.   3. Check if user has modified any information.   4. Update player’s info in database.   5. Notify user with message, “Player’s information updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select player from list of all players in team. 2. User update player’s information 3. Clicks on update player’s details button 4. Repeat step 1 to 3. | * 1. Get selected player’s information.   2. Show that player’s information.   3. Check if user has modified any information.   4. Notify user with message, “To update player, the information must have been changed!” |
| Post conditions | Team manager has successfully updated player’s information. | |
| Exception  Conditions | Team manager is not allowed to update player without details like player’s image, first name, last name, age and position | |

**Delete Player**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-12 | |
| Use Case Name | Delete Player | |
| Date created | 22/05/020 | |
| Actor | Team Manager | |
| Description | Team manager can delete player. | |
| Triggering event | Team manager clicks on delete button. | |
| Preconditions | Team manager must be logged into system. | |
| Flow of events | Actor | System |
| 1. User select player from list of all players in team. 2. Clicks on delete button | * 1. Get selected player’s information.   2. Show that player’s information.   3. Check if player is not in any ongoing league.   4. Delete player from database.   5. Notify user with message, “Player’s deleted successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select player from list of all players in team. 2. Clicks on delete button 3. Repeat step 1 and 2 | * 1. Get selected player’s information.   2. Show that player’s information.   3. Check if player is not in any ongoing league.   4. Notify user with message, “You cannot delete this player!” |
| Post conditions | Team manager has successfully deleted player. | |
| Exception  Conditions | Team manager is not allowed to delete player if player is in ongoing league. | |

* **Create Schedule**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-13 | |
| Use Case Name | Create Schedule | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can set match schedule for specific league. | |
| Triggering event | League manager clicks on Create match schedule button. | |
| Preconditions | League manager must have logged in and added at least one league and that league contains at least 2 teams. | |
| Flow of events | Actor | System |
| 1. User clicks on create schedule button from league features screen. 2. User Select date. 3. User Select time. 4. User enters Location. 5. User Select team 1. 6. User Select team 2. 7. Clicks on create match button. | * 1. System verifies data which is entered by user.   2. Save schedule’s info in database.   3. Notify user with message “Schedule added successfully!” |
| Alternative flow of events | Actor | System |
| 1. User clicks on create schedule button from league features screen. 2. User Select date. 3. User Select time. 4. User enters Location. 5. User Select team 1. 6. User Select team 2. 7. Clicks on create match button. 8. Repeat step 1 to 7. | * 1. System verifies data which is entered by user.   2. Notify user with message “Team already has a match on selected date or time!” |
| Post conditions | League manager has successfully added schedule. | |
| Exception  Conditions | League manager is not allowed to add schedule without date, time, place or teams. | |

* **Update Shedule**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-14 | |
| Use Case Name | Update Schedule | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can update match schedule for specific league. | |
| Triggering event | League manager clicks on update match schedule button. | |
| Preconditions | League manager must have logged in and added at least one league and that league contains at least 2 teams. | |
| Flow of events | Actor | System |
| 1. User select league from all leagues. 2. User select already added schedule. 3. User Select date. 4. User Select time. 5. User enters Location. 6. User Select team 1. 7. User Select team 2. 8. Clicks on update schedule button. | * 1. Get selected schedule’s information.   2. Show that schedule’s information.   3. Check if match is live or not.   4. Notify user with message, “Schedule updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select league from all leagues. 2. User select already added schedule. 3. User Select date. 4. User Select time. 5. User enters Location. 6. User Select team 1. 7. User Select team 2. 8. Clicks on update schedule button. 9. Repeat step 1 to 8. | * 1. Get selected schedule’s information.   2. Show that schedule’s information.   3. Check if match is live or not.   4. Notify user with message, “You cannot update live match’s schedule!” |
| Post conditions | League manager has successfully updated schedule. | |
| Exception  Conditions | League manager is not allowed to update live match’s schedule. | |

**Delete Schedule**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-15 | |
| Use Case Name | Delete Schedule | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can delete match schedule for specific league. | |
| Triggering event | League manager clicks on delete match schedule button. | |
| Preconditions | League manager must have logged in and already added scheduled match. | |
| Flow of events | Actor | System |
| 1. User select league from all leagues. 2. User select already added schedule. 3. Clicks on delete schedule button. | * 1. Get selected schedule’s information.   2. Show that schedule’s information.   3. Check if match is live or not.   4. Notify user with message, “Schedule deleted successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select league from all leagues. 2. User select already added schedule. 3. Clicks on delete schedule button. 4. Repeat step 1 to 3. | * 1. Get selected schedule’s information.   2. Show that schedule’s information.   3. Check if match is live or not.   4. Notify user with message, “You cannot delete live match’s schedule!” |
| Post conditions | League manager has successfully updated schedule. | |
| Exception  Conditions | League manager is not allowed to delete live match’s schedule. | |

* **Add Or Update Schedule**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-16 | |
| Use Case Name | Add or update Schedule | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can add or update score for match. | |
| Triggering event | League manager clicks on update score button. | |
| Preconditions | League manager must have logged in. | |
| Flow of events | Actor | System |
| 1. User select league. 2. User select played match. 3. User enters number of goals. 4. User enters number of fouls. 5. User enters number of corners. 6. User enters number of free kicks. 7. User enters number of goals saved. 8. Click on Update score button. | * 1. Check if any value is empty or not.   2. Notify user with message, “Score updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select league. 2. User select played match. 3. User enters number of goals. 4. User enters number of fouls. 5. User enters number of corners. 6. User enters number of free kicks. 7. User enters number of goals saved. 8. Click on Update score button. | * 1. Check if any value is empty or not.   2. Notify user with message, “Required fields are missing!” |
| Post conditions | League manager has successfully added or updated score. | |
| Exception  Conditions | League manager is not allowed to update score for upcoming match schedule. | |

* **Add Or Update Schedule**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-16 | |
| Use Case Name | Add or update Schedule | |
| Date created | 22/05/2020 | |
| Actor | League Manager | |
| Description | League manager can add or update score for match. | |
| Triggering event | League manager clicks on update score button. | |
| Preconditions | League manager must have logged in. | |
| Flow of events | Actor | System |
| 1. User select league. 2. User select played match. 3. User enters number of goals. 4. User enters number of fouls. 5. User enters number of corners. 6. User enters number of free kicks. 7. User enters number of goals saved. 8. Click on Update score button. | * 1. Check if any value is empty or not.   2. Notify user with message, “Score updated successfully!” |
| Alternative flow of events | Actor | System |
| 1. User select league. 2. User select played match. 3. User enters number of goals. 4. User enters number of fouls. 5. User enters number of corners. 6. User enters number of free kicks. 7. User enters number of goals saved. 8. Click on Update score button. | * 1. Check if any value is empty or not.   2. Notify user with message, “Required fields are missing!” |
| Post conditions | League manager has successfully added or updated score. | |
| Exception  Conditions | League manager is not allowed to update score for upcoming match schedule. | |

* **View Teams**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-17 | |
| Use Case Name | View Teams | |
| Date created | 22/05/2020 | |
| Actor | League Manager, Team managers and Guest users. | |
| Description | View teams page allow user to see all teams in a league. | |
| Triggering event | User has to select league from league list. | |
| Preconditions | League manager must has logged in. | |
| Flow of events | Actor | System |
| 1. User select league from home page. 2. User select team from team list. | * 1. Get and display all teams.   2. Display selected team info from database. |
| Alternative flow of events | Actor | System |
| 1. User select league from home page. 2. User select team from team list. | * 1. Get and display all teams.   2. Notify User with message, “Could not load details please try again”. |
| Post conditions | User can see all team list and team details. | |
| Exception Conditions | Could not load details due to some issues. | |

* **View Leagues**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-18 | |
| Use Case Name | View Leagues | |
| Date created | 22/05/2020 | |
| Actor | League Manager, Team managers and Guest Users. | |
| Description | View League page allow users to see all leagues. | |
| Triggering event | Users can view all leagues on home page | |
| Preconditions | League manager must have logged in. | |
| Flow of events | Actor | System |
| 1. User select leagues tab on home page. | * 1. Get and display all leagues list from database. |
| Alternative flow of events | Actor | System |
| 1. User select leagues tab on home page. | * 1. Notify User with message, “Could not load details please try again”. |
| Post conditions | User can see all league details. | |
| Exception Conditions | Could not load details due to some issues. | |

* **View Players**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-19 | |
| Use Case Name | View Players | |
| Date created | 22/05/2020 | |
| Actor | League Manager, Team Manager and Guest Users | |
| Description | All user can see player details | |
| Triggering event | Users clicks on any player from team player’s list. | |
| Preconditions |  | |
| Flow of events | Actor | System |
| 1. User select league from league list on home page. 2. User select team 3. User select player | 1. Display List of Teams 2. Display list of players in team    1. Get and display all player info from database. |
| Alternative flow of events | Actor | System |
| 1. User select league from league list on home page. 2. User select team 3. User select player | 1. Display List of Teams 2. Display list of players in team    1. Notify User with message, “Could not load details please try again”. |
| Post conditions | User can see all details about player. | |
| Exception Conditions | Could not load details due to some issues. | |

* **View Score Board**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-20 | |
| Use Case Name | View Score Board | |
| Date created | 22/05/2020 | |
| Actor | League Manager, Team Manager and Guest Users | |
| Description | All users can see score board on score board page. | |
| Triggering event | Users clicks on scoreboard page | |
| Preconditions |  | |
| Flow of events | Actor | System |
| 1. User select scoreboard tab on home page. | * 1. Get and display all scores of played matches. |
| Alternative flow of events | Actor | System |
| 1. User select scoreboard tab on home page. | * 1. Notify User with message, “Could not load details please try again”. |
| Post conditions | User can see score of played matches. | |
| Exception Conditions | Could not load details due to some issues. | |

* **View Upcoming Messages**

|  |  |  |
| --- | --- | --- |
| Use Case ID | UC-21 | |
| Use Case Name | View Upcoming matches | |
| Date created | 22/05/2020 | |
| Actor | League Manager, Team manager and Guest Users. | |
| Description | All users can see upcoming matches, team details, date, time and match place. | |
| Triggering event | User clicks on upcoming match button. | |
| Preconditions |  | |
| Flow of events | Actor | System |
| 1. User select upcoming match tab on home page. | * 1. Get and display list of all upcoming matches. |
| Alternative flow of events | Actor | System |
| 1. User select upcoming match tab on home page. | * 1. Notify User with message, “Could not load details please try again”. |
| Post conditions | User can see all upcoming match detail like match place, date, time and player in team. | |
| Exception Conditions | Could not load details due to some issues. | |

* 1. **Class Diagram**

A close up of a map

Description automatically generated

# **Entity \_relationship Diagram**

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**

A close up of a logo

Description automatically generated

## **5.2Logical schema of firebase database**

A screenshot of a cell phone

Description automatically generated

## **5.3Database Script**

DROP SCHEMA IF EXISTS `soccer\_app` ;

-- -----------------------------------------------------

-- Schema soccer\_app

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `soccer\_app` DEFAULT CHARACTER SET utf8 COLLATE utf8\_bin ;

USE `soccer\_app` ;

-- -----------------------------------------------------

-- Table `soccer\_app`.`user`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`user` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`user` (

`user\_id` INT(11) NOT NULL AUTO\_INCREMENT,

`first\_name` VARCHAR(45) NOT NULL,

`last\_name` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`user\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`leagues`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`leagues` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`leagues` (

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

`league\_name` VARCHAR(45) NOT NULL,

`league\_manager\_id` INT(11) NOT NULL,

`league\_icon` BLOB NULL,

PRIMARY KEY (`league\_id`),

INDEX `league\_manager\_idx` (`league\_manager\_id` ASC) VISIBLE,

CONSTRAINT `league\_manager`

FOREIGN KEY (`league\_manager\_id`)

REFERENCES `soccer\_app`.`user` (`user\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`teams`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`teams` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`teams` (

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

`team\_name` VARCHAR(45) NOT NULL,

`team\_icon` BLOB NULL DEFAULT NULL,

`team\_manager\_id` INT(11) NOT NULL,

`team\_manager\_contact` VARCHAR(45) NOT NULL,

PRIMARY KEY (`team\_id`),

INDEX `team\_manager\_idx` (`team\_manager\_id` ASC) VISIBLE,

CONSTRAINT `team\_manager`

FOREIGN KEY (`team\_manager\_id`)

REFERENCES `soccer\_app`.`user` (`user\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`league\_teams`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`league\_teams` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`league\_teams` (

`idleague\_teams` INT(11) NOT NULL AUTO\_INCREMENT,

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

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

PRIMARY KEY (`idleague\_teams`),

INDEX `league\_ref\_idx` (`league\_id` ASC) VISIBLE,

INDEX `team\_ref\_idx` (`team\_id` ASC) VISIBLE,

CONSTRAINT `league\_ref`

FOREIGN KEY (`league\_id`)

REFERENCES `soccer\_app`.`leagues` (`league\_id`),

CONSTRAINT `team\_in\_league\_ref`

FOREIGN KEY (`team\_id`)

REFERENCES `soccer\_app`.`teams` (`team\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`match\_schedules`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`match\_schedules` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`match\_schedules` (

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

`match\_location` VARCHAR(45) NOT NULL,

`match\_date` DATE NOT NULL,

`match\_time` DATETIME NOT NULL,

`team1\_id` INT(11) NOT NULL,

`team2\_id` INT(11) NOT NULL,

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

PRIMARY KEY (`match\_schedule\_id`),

INDEX `match\_in\_league\_ref\_idx` (`league\_id` ASC) VISIBLE,

INDEX `team1\_in\_match\_ref\_idx` (`team1\_id` ASC) VISIBLE,

INDEX `team2\_in\_match\_ref\_idx` (`team2\_id` ASC) VISIBLE,

CONSTRAINT `match\_in\_league\_ref`

FOREIGN KEY (`league\_id`)

REFERENCES `soccer\_app`.`leagues` (`league\_id`),

CONSTRAINT `team1\_in\_match\_ref`

FOREIGN KEY (`team1\_id`)

REFERENCES `soccer\_app`.`teams` (`team\_id`),

CONSTRAINT `team2\_in\_match\_ref`

FOREIGN KEY (`team2\_id`)

REFERENCES `soccer\_app`.`teams` (`team\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`match\_scores`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`match\_scores` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`match\_scores` (

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

`match\_id` INT(11) NOT NULL,

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

`goals` INT(11) NOT NULL,

`fouls` INT(11) NOT NULL,

`corners` INT(11) NOT NULL,

`free\_kicks` INT(11) NOT NULL,

`goal\_saved` INT(11) NOT NULL,

PRIMARY KEY (`match\_scores\_id`),

INDEX `score\_of\_match\_ref\_idx` (`match\_id` ASC) VISIBLE,

INDEX `score\_of\_tem\_ref\_idx` (`team\_id` ASC) VISIBLE,

CONSTRAINT `score\_of\_match\_ref`

FOREIGN KEY (`match\_id`)

REFERENCES `soccer\_app`.`match\_schedules` (`match\_schedule\_id`),

CONSTRAINT `score\_of\_tem\_ref`

FOREIGN KEY (`team\_id`)

REFERENCES `soccer\_app`.`teams` (`team\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`players`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`players` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`players` (

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

`player\_first\_name` VARCHAR(45) NOT NULL,

`player\_last\_name` VARCHAR(45) NOT NULL,

`player\_age` INT(11) NOT NULL,

`player\_image` BLOB NULL DEFAULT NULL,

`player\_position` VARCHAR(45) NOT NULL,

PRIMARY KEY (`player\_id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

-- -----------------------------------------------------

-- Table `soccer\_app`.`team\_players`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `soccer\_app`.`team\_players` ;

CREATE TABLE IF NOT EXISTS `soccer\_app`.`team\_players` (

`idteam\_players` INT(11) NOT NULL AUTO\_INCREMENT,

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

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

PRIMARY KEY (`idteam\_players`),

INDEX `player\_ref\_idx` (`player\_id` ASC) VISIBLE,

INDEX `team\_ref\_idx` (`team\_id` ASC) VISIBLE,

CONSTRAINT `player\_ref`

FOREIGN KEY (`player\_id`)

REFERENCES `soccer\_app`.`players` (`player\_id`),

CONSTRAINT `team\_ref`

FOREIGN KEY (`team\_id`)

REFERENCES `soccer\_app`.`teams` (`team\_id`))

ENGINE = InnoDB

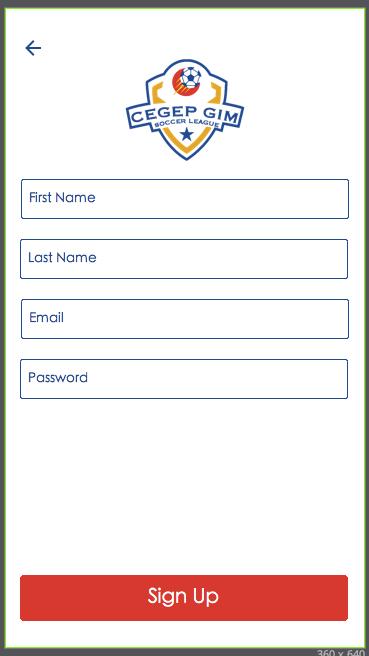
DEFAULT CHARACTER SET = utf8

COLLATE = utf8\_bin;

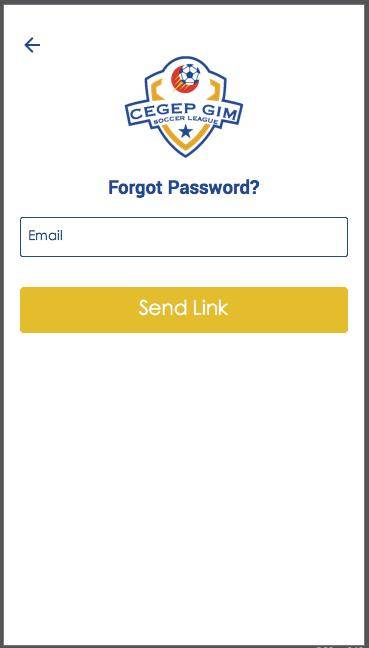
## **Screen Design**

* **Launch screen, sign up, login and forgot password**

A picture containing fruit

Description automatically generated 

A screenshot of a cell phone

Description automatically generated 

* **Scoreboard tab screen (on Home page)**

A screenshot of a cell phone

Description automatically generated 

* **Leagues tab screen (on Home page)**

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

* **Upcoming matches tab screen (on Home page)**

 A screenshot of a cell phone

Description automatically generated

* **Side menu for all users**

For guest users

For league manager

For team manager

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

* **League manager features**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

(This is scrollable screen)

* **League manager schedule screens**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

* **League manager scoreboard screens**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

* **Team manager screen for team manager**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

* **Team players screen for team manager**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

* **Update profile screen for league manager and team manager**

A screenshot of a cell phone

Description automatically generated