# RecDawgs

Recreational Sports Management System

Version 1.0

1 February 2016

Team9

Justin Tumale

Logan Jahnke

Jay Springfield

Bowen Yang

# Table of Contents

# 1 - Introduction

## 1.1 - Purpose

RecDawgs is a Recreational Sports Management Systems designed to support the management of several recreational sports leagues for a small college with minimal involvement of college staff.

## 1.2 - Scope

The product to be produced is titled 'RecDawgs'. The scope of RecDawgs includes its features, benefits, and limitations.

**1.2.1 - Features:**

1. The system will allow an administrator to enter into the system a number of *sports venues*. Each sports venue will specify its *name*, *address*, and if it is an *indoor* or *outdoor* venue.
2. The system will allow an administrator to enter into the system a number of *sports leagues*, for example, touch football, softball, or tennis. An administrator will be able to specify the *official league and match rules* for each league. Also, the administrator will be able to define the *minimum and maximum number of teams*, as well as the *minimum and maximum number of team members*. A league will also be specified as *indoor* or *outdoor*.
3. An administrator will be able to *make changes* to any information currently stored in the system. For example, it will be possible to change the minimum/maximum number of teams, league and match rules, re-assign sports venues to different leagues, etc. It will be possible to remove leagues, venues, etc., as well.
4. A user (a student) will be able to *register with the system*. To do that, the user will establish the *username* and *password*, and then provide a *student number*, a *college major, email address*, and the *residence address*. A registered user will be able to login and logout from the system. A user will be able to *modify* his/her user information.
5. A user will be able to create a *team* and enter it for one of the available sports leagues, provided the current number of teams is lower than the defined maximum for the league. The user specifies the *name of the team*. Furthermore, the user who created a team automatically becomes the *captain of the team*.
6. A user will be able to sign up for one of the existing teams, provided the current number of team members is lower than the defined maximum for the league. Once the team

reaches the minimum number of team members, it becomes a full participant in the league.

7. A sports league, which does not have the minimum number of fully participating teams, as defined for the league, may be *cancelled* by the administrator. Otherwise, it becomes *active*. Teams in an active league play *matches*. Each team plays one match against each other team in the league. Each match will have a winner (no draws are allowed) and the *score* will be numeric (two non-negative numbers).
8. The system will be able to automatically create a list of matches to be played in league, once it becomes active, listing the opposing teams for each match. A match has *two teams* and a *sports venue*. The list is subdivided into rounds.
9. Both team captains will enter the same *score of a match* for it to be official and recorded in the league. In case of a dispute, an administrator will also be able to enter the score of a match. Once all matches have been played, one team is designated as the *winner of the league*. An administrator will also be able to indicate the league winner, if needed.
10. A user, even a team's captain, will be able to cancel his/her membership from a team. In case there are unexpected changes to the team roster, an administrator will be able to appoint a regular team member as the team's captain.
11. A user will be able to select a league and view the match results and a current table of teams, including a summary of wins, losses, and a cumulative score (all points won and lost).
12. The system will be *accessible from a common Web browser* (such as the Mozilla Firefox, Google Chrome, Safari, and Internet Explorer).
13. The system should provide *multi-user access*, assuring correct concurrent behavior. The system should maintain suitable *authorization* information and *authenticate access*. *User authentication* should be implemented by checking *username* and *password*.
14. The system will have an *easy-to-use user interface* (UI) with screens designed for each part of the system's functionality and suitable for different types of users (customers, administrators, managers).
15. The system will use a *persistent data store* (MySQL RDBMS) for all of the relevant data.
16. The system should use accepted *standards* whenever possible (HTML, CGI, Servlet API, JDBC, ODBC, SQL, etc.). The project *will* be coded in either C++ or Java, possibly including other scripting languages, such as PHP and JavaScript, if needed.

### 1.2.3 - Benefits:

The benefits of RecDawgs is to provide organization of sports league competitions with ease of use and minimal involvement of college staff.

**1.2.4 - Limitations:**

RecDawgs will not:

- keep a live score of ongoing matches.
- feature highlight videos of league matches.
- allow users to sign up to multiple teams or leagues, due to the high risk of scheduling conflicts.
- offer a direct messaging service.
- offer a bulletin posting service.
- allow any other user besides the administrator to assign team captains.

## 1.3 - Definitions, Acronyms, and Abbreviations

## 1.4 - References

Almstrum, V., Dr. (n.d.). Software Requirements Document Model. Retrieved February 1, 2016, from http://web.stonehill.edu/compsci/CS400/SoftwareRequirementsDocument.doc

Kochut, K. J., Dr. (2016). Term Project. Retrieved February 01, 2016, from http://cobweb.cs.uga.edu/~kochut/teaching/x050/TermProject.html

McKinnon, A. D. (2005, February 09). Software Requirements Specification Template. Retrieved February 1, 2016, from http://www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

## 1.5 - Overview

This requirements document for RecDawgs contains the introduction, a specification for the proposed system including the requirements, constraints, and system models, and a glossary. It is organized as such:

Section 1: Introduction

1.1 an introduction to the RecDawgs system

1.2 the scope of RecDawgs, which includes its features, benefits, and limitations.

1.3 a definition of terms, acronyms, and abbreviations that will be used throughout this document

1.4 the limitations of RecDawgs

Section 2: Current system (skip)

# 2 - Current System (Skip)

# 3 - Proposed System

## 3.1 - System Overview

The proposed system, RECDAWGS, is our solution to manage recreational sports at a college. The system, which will use a website and a database, will allow an administrator to facilitate things like creating and managing leagues, monitoring game scores, and monitoring site activities in general. Players will be allowed to create teams, join teams, and play matches against other teams in the league based on a schedule laid out by the RECDAWGS system. Each team will have a team captain, who will manage their team and report scores from the matches played throughout the season. All users of the RECDAWGS system will be able to access the system by registering, logging in, view league stats, and log out of the system.

## 3.2 - Functional Requirements

### 3.2.1 - Administrator Interactions:

- Create League
- Edit League
- Cancel League
- Create Sports Venue
- Appoint Team Captain
- Report Match
- Logout

### 3.2.2 - Player Interactions:

- Create Team
- Join Team
- Leave Team
- Logout

### 3.2.3 - Captain Interactions:

- Leave Team
- Report Match
- Logout

### 3.2.4 - Unauthenticated User Interactions:

- Register
- Login

## 3.3 - Non-functional Requirements

### 3.3.1 - Performance:

- RecDawgs will provide multi-user access (Administrator and Player)
- RecDawgs will have an easy-to-use software interface for both the Administrator(s) and the Player(s)
- RecDawgs will have persistent data storage using MySQL RDBSM
- RecDawgs will be accessible from any common internet browser
  - Safari
  - Google Chrome
  - Firefox
  - Internet Explorer

### 3.3.2 - Reliability:

- RecDawgs will be available 24 hours/day and 7 days/week

### 3.3.3 - Maintainability:

- The Administrator will have complete control over Leagues and Teams

### 3.3.4 - Implementation:

- RecDawgs will be implemented using the standard programming languages.
  - HTML/CSS
  - SQL
  - Javascript

### 3.3.5 - Extensibility:

- Administrators will be able to add new Sports Venues and new Leagues
- Players will be able to create teams

## 3.4 - Constraints

- The implementation language must be standard programming languages
- The user-interface must be accessible from any common internet browser

## 3.5 - System Models

### 3.5.1 - Scenarios:

| Scenario | **Dan Creates a team with his friends** |
|---|---|
| Participating Actors | **Team Captain**: Dan<br>**Players:** Dan's friends |
| Steps | 1. Dan and his friends would like to play in the RECDAWGS soccer league as a new team.<br>2. Dan registers for RECDAWGS.<br>3. Dan logs into RECDAWGS.<br>4. Seeing that there are still available slots for new teams in the soccer league, Dan enters the League Team menu and presses 'CREATE NEW TEAM'.<br>5. Dan enters the team's name and hits submit and is by default the 'Team Captain'.<br>6. Dan's friends also register for RECDAWGS.<br>7. They select Dan's team from the League Team menu and join his team by pressing 'JOIN TEAM'. |

| Scenario | **Andrew wants to join the system through registration.** |
|---|---|
| Participating Actors | **Player:** Andrew |
| Steps | 1. Andrew, a UGA student, wants to play an IM sport.<br>2. Andrew presses the "Register" button on the website.<br>3. The RECDAWGS system presents Andrew with a registration form with empty text boxes for information like email address, physical address, phone number, user name, and password.<br>4. Andrew fills out the information and presses the "Save and Register" |

| | |
|---|---|
| | button.<br>5. Andrew receives an email to verify his registration with the system.<br>6. Andrew is now in the system and able to either join a team, or create a team. |

| | |
|---|---|
| Scenario | **Bob wants to see how his friend's league is doing** |
| Participating Actors | **Player:** Bob |
| Steps | 1. Bob wants to see how well his friend's team is doing<br>2. Bob logs into RECDAWGS<br>3. Bob finds his friends league and clicks on 'View League Stats'<br>4. RECDAWGS displays the current scores for the league his friend is playing in |

| | |
|---|---|
| Scenario | **Jay wants to change the name of a league.** |
| Participating Actors | **Admin:** Jay |
| Steps | 1. Jay is an admin and sees that he accidently spelled a league name incorrectly<br>2. Jay logs into his Admin account<br>3. Jay presses the "Edit League" button<br>4. Jay is presented with a form with the league's current information already filled in<br>5. Jay clicks on the league name text box, and types in the correct name<br>6. Jay presses the "Save Changes" button and the form screen closes |

| | |
|---|---|
| Scenario | **Sarah spelled her name wrong in the system** |
| Participating Actors | **Player:** Sarah |
| Steps | 1. Sarah recently registered for RECDAWGS<br>2. Sarah notices that her name is entered as ‘Saeah’ in the system<br>3. Sarah logs into her RECDAWGS account and clicks ‘Modify Account’ from the main menu<br>4. Sarah clicks on the ‘Name’ field and changes it to ‘Sarah’<br>5. Sarah presses submit and her account name is changed |

| | |
|---|---|
| Scenario | **John (Team captain) leaves the team** |
| Participating Actors | **Team Captain:** John<br>**Admin**: Sarah |
| Steps | 1. John decides he doesn’t have time to play soccer.<br>2. John presses the ‘Leave team’ button.<br>3. Sarah is notified that a team captain (John) has left his/her team.<br>4. Sarah appoints a new captain by pressing the ‘Appoint Captain’ button.<br>5. Sarah fills out the form and chooses the new captain<br>6. The team players are notified of the change. |

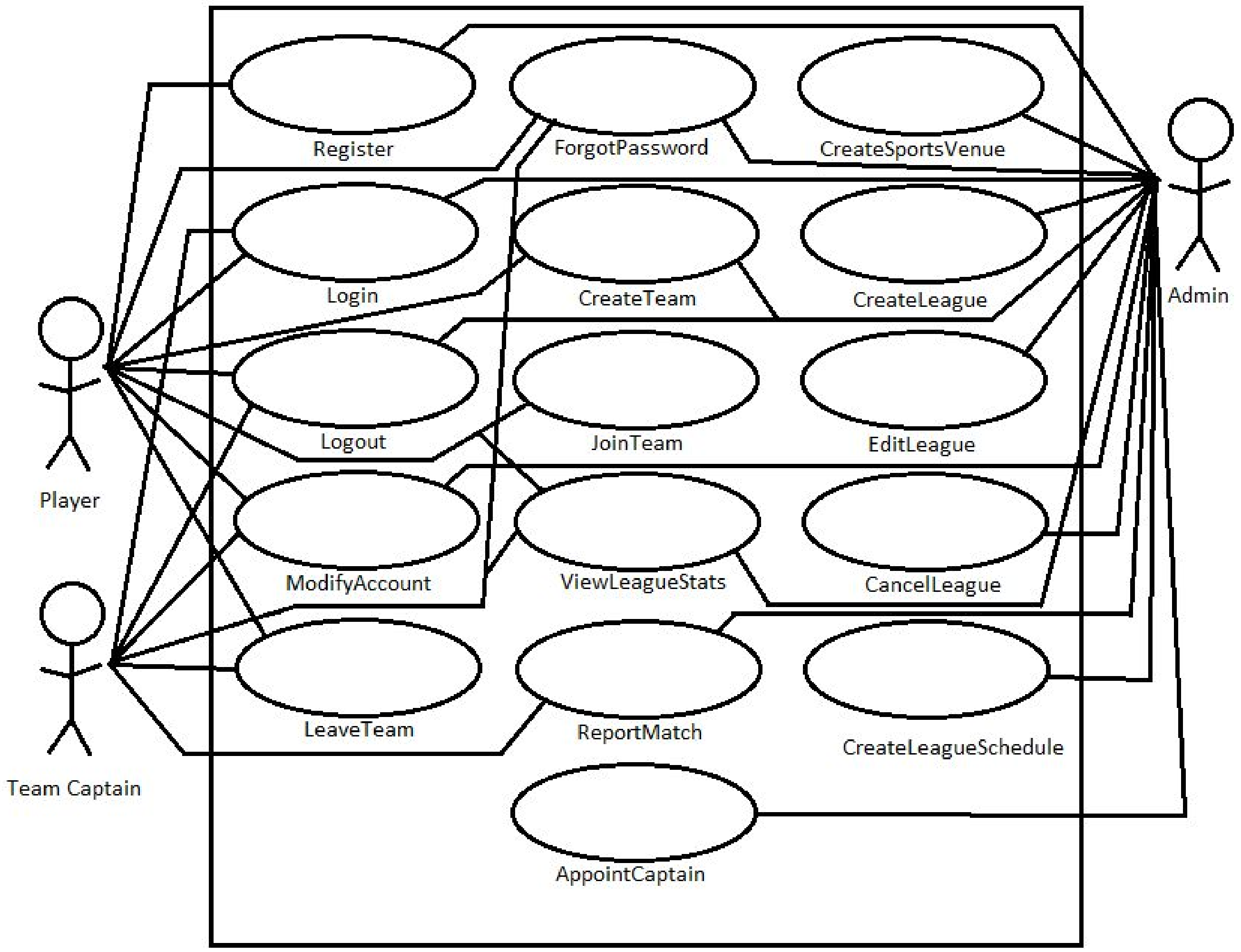
| | |
|---|---|
| Scenario | **Team loses a match** |
| Participating Actors | **Team Captain:** Bob & John<br>**Admin:** Sarah |
| Steps | 1. Bob's team loses a match against John's team.<br>2. Bob reports the score of the match to RECDAWGS.<br>3. John reports the score of the match to RECDAWGS.<br>4. RECDAWGS compares reports. If they are the same, the match is put into the database and the League Stats are updated; else, RECDAWGS notifies the admin of the conflict and the Admin reports the final score of the match. |

| | |
|---|---|
| Scenario | **Bob wants to play Soccer, but it’s not registered in RECDAWGS** |
| Participating Actors | **Player:** Bob<br>**Admin:** Seth |
| Steps | 1. Bob checks RECDAWGS and was unable to find Soccer under Leagues<br>2. Bob talks to Seth and asks Seth to add Soccer<br>3. Seth logs into his admin account<br>4. Seth clicks on the Create League button and fills out the necessary information<br>5. Bob logs in and signs up for Soccer |

| | |
|---|---|
| Scenario | **A bad snowstorm is predicted to occur during a league** |
| Participating Actors | **Admin:** Seth |
| Steps | 1. Seth finds out that a snowstorm is going to happen during the set time for a league<br>2. Seth logs in to his Admin account<br>3. Seth clicks on Cancel League button<br>4. All players in the league are notified of the change |

## 3.5.2 Use Case model

### 3.5.2.1 - Use Case Model Diagram:



### 3.5.2.2 - Use Cases:

#### 3.5.2.2.1 - CreateSportsVenue

| | |
|---|---|
| Name | Create Sports Venue |
| ID | CreateSportsVenue |
| Version | 1.0 |
| Author | Logan Jahnke |
| Date | 01/27/2016 |
| Summary | Use case to create a new **SportsVenue** that can be used for games in the leagues. |
| Basic Path | 1. The **Admin** presses the "Create Sports Venue" button.<br>a. **RECDAWGS** responds by presenting a form to the Admin.<br>2. The **Admin** completes form that specifies the location of the venue, whether it is indoor or outdoor, and its name.<br>a. **RECDAWGS** receives form, compares it database to check if duplication exists.<br>b. **RECDAWGS** adds the **SportsVenue** to the database. |
| Alternative Paths | 1. The **Admin** presses the cancel button, the creation of the **SportsVenue** ceases, and the Use Case is terminated. |
| Exception Paths | 1. If **RECDAWGS** in step 2a finds a duplicate, then the action is canceled, and the **Admin** is notified that he/she made a duplicate. |
| Extension Points | |
| Triggers | The **Admin** presses the “Create Sports Venue” button. |
| Assumption | |
| Pre-conditions | The **Admin** is logged into **RECDAWGS**. |
| Post-conditions | The **SportsVenue** object is added to the database. |

### 3.5.2.2.2 - CreateLeague

| | |
|---|---|
| Name | Create League |
| ID | CreateLeague |
| Version | 1.0 |
| Author | Logan Jahnke, Bowen Yang |
| Date | 01/27/2016 |
| Summary | Use case to create a new **League** which host a variety of teams. |
| Basic Path | 1. The **Admin** presses the "Create League" button.<br>a. **RECDAWGS** responds by presenting a form to the **Admin**.<br>2. The **Admin** completes form that specifies any initial teams that are in the league, as well as any initial players on each team.<br>a. **RECDAWGS** receives form, adds it to the database, and sends confirmation to admin. |
| Alternative Paths | 1. The **Admin** presses cancel and the Use Case is terminated. |
| Exception Paths | 1. If RECDAWGS in step 2 finds another league of the same name. A error message will pop up and ask the admin whether he/she will rename |
| Extension Points | |
| Triggers | The **Admin** presses the “Create Sports Venue” button. |
| Assumption | |
| Pre-conditions | The **Admin** is logged into **RECDAWGS**. |
| Post-conditions | The **SportsVenue** object is added to the database. |

**3.5.2.2.3 - EditLeague**

| | |
|---|---|
| Name | Edit League |
| ID | EditLeague |
| Version | 1.0 |
| Author | Logan Jahnke, Bowen Yang |
| Date | 01/28/2016 |
| Summary | Use case to edit an existing **League**. |
| Basic Path | 1. The **Admin** presses the "Edit League" button.<br>    a. **RECDAWGS** responds by presenting a form to the admin that already has all the information about the **League** being edited that's currently stored.<br>2. The **Admin** edits form that specifies all information about the **League** that was already there.<br>    a. **RECDAWGS** receives edited form, adds it to the database, and sends confirmation of edits **Admin**, as well as **Player(s)** within that **League**. |
| Alternative Paths | 1. The **Admin** presses the cancel button, the edit of the **League** ceases, and the Use Case is terminated. |
| Exception Paths | 1. If the edit conflicts with current conditions, the **Admin** will be presented with a message “Requirements conflict with current conditions”. i.e. if 10 teams are signed up but you change limit to 8. |
| Extension Points | |
| Triggers | The **Admin** presses the “Edit League” button. |
| Assumption | There is a **League** to edit. |
| Pre-conditions | The **Admin** is logged into **RECDAWGS**. |
| Post-conditions | The **League** object is edited in the database. |

### 3.5.2.2.4 - Register

| | |
|---|---|
| Name | Register |
| ID | Register |
| Version | 1.0 |
| Author | Logan Jahnke, Bowen Yang |
| Date | 01/28/2016 |
| Summary | Use case to create a new **User** who can join teams to become a **Player** or create a team to become a **Captain**. |
| Basic Path | 1. The **User** presses the "Register" button.<br>a. **RECDAWGS** responds by presenting a form to the **User**.<br>2. The **User** completes form that asks for information like a user name, password, a student number, a major, email address, and residence address.<br>a. **RECDAWGS** receives form and sends a confirmation email to the **User** so that they can be verified. |
| Alternative Paths | 1. The **User** presses the cancel button, the creation of the **User** ceases, and the Use Case is terminated. |
| Exception Paths | 1. If the **User** does not fill out a text field (Username, Password, etc.), **RECDAWGS** will represent the form and the Use Case will fallback to step 2.<br>2. If the email field is already in the database, **RECDAWGS** will represent the form, notify the **User** that the email is in use, and the Use Case will fallback to step 2.<br>3. If player name is already taken, the player will be prompt to change his display name |
| Extension Points | |
| Triggers | The **User** presses the “Register” button. |
| Assumption | |
| Pre-conditions | |
| Post-conditions | The **User** object is added to the database. |

**3.5.2.2.5 - Login**

| | |
|---|---|
| Name | Login |
| ID | Login |
| Version | 1.0 |
| Author | Justin Tumale |
| Date | 01/28/2016 |
| Summary | Use case to log a **User** or **Admin** into their account. |
| Basic Path | 1. **RECDAWGS** displays text entry boxes for Username and Password.<br>2. **User**/**Admin** enters valid Username and Password.<br>3. **User**/**Admin** presses "Login" button located below text entry fields.<br>4. **RECDAWGS** verifies that the **User**/**Admin** has entered valid Username and Password.<br>5. **User**/**Admin** is granted access to **RECDAWGS**. |
| Alternative Paths | 1. The **User**/**Admin** presses the cancel button, the login procedure ceases, and the Use Case is terminated. |
| Exception Paths | 1. If the **User**/**Admin** types in the wrong Username or wrong Password, **RECDAWGS** notifies the **User**/**Admin**, the form is represented, and the Use Case will fallback to step 1. |
| Extension Points | |
| Triggers | The **User**/**Admin** presses the "Login" button. |
| Assumption | The **User**/**Admin** already has an account. |
| Pre-conditions | The **User**/**Admin** is not logged into **RECDAWGS**. |
| Post-conditions | The **User**/**Admin** is logged into **RECDAWGS**. |

**3.5.2.2.6 - Logout**

| | |
|---|---|
| Name | Logout |
| ID | Logout |
| Version | 1.0 |
| Author | Justin Tumale |
| Date | 01/28/2016 |
| Summary | Use case to logout a **User**/**Admin**. |
| Basic Path | 1. **User**/**Admin** navigates to their respective Main Menu.<br>2. **User**/**Admin** presses the "Logout" button on the upper left hand side of the screen.<br>3. **User**/**Admin** exits **RECDAWGS** and they are redirected back to the Login page. |
| Alternative Paths | N/A |
| Exception Paths | |
| Extension Points | |
| Triggers | The **User**/**Admin** presses the "Logout" button. |
| Assumption | |
| Pre-conditions | The **User**/**Admin** is logged into **RECDAWGS**. |
| Post-conditions | The **User**/**Admin** is not logged into **RECDAWGS**. |

**3.5.2.2.7 - ModifyAccount**

| | |
|---|---|
| Name | Modify Account |
| ID | ModifyAccount |
| Version | 1.0 |
| Author | Justin Tumale |
| Date | 01/29/2016 |
| Summary | Use case to edit a **User**. |
| Basic Path | 1. The **User** presses the “Modify Account” button from the Main Menu.<br>2. **RECDAWGS** brings the **User** to the Modify Account menu.<br>3. The **User** makes the necessary changes to the fields that are provided in the Modify Account menu.<br>4. The **User** presses the “Submit” button, and is taken back to the Main Menu. |
| Alternative Paths | In step 3, instead of making changes, the PLAYER decides to discard the changes by pressing the 'BACK' button.<br><br>In step 3 or 4, instead of submitting the changes, the PLAYER decides to discard the changes by pressing the 'BACK' button. |
| Exception Paths | 1. If the email field is already in the database, **RECDAWGS** will represent the form, notify the **User** that the email is in use, and the Use Case will fallback to step 2. |
| Extension Points | |
| Triggers | The PLAYER would like to update his or her current account information. |
| Assumption | |
| Pre-conditions | The **User** is logged into **RECDAWGS**. |
| Post-conditions | The **User** object is edited in the database. |

**3.5.2.2.8 - ForgotPassword**

| | |
|---|---|
| Name | Forgot Password |
| ID | ForgotPassword |
| Version | 1.0 |
| Author | Jerry Springfield Jr., Logan Jahnke |
| Date | 01/28/2016 |
| Summary | Use case to assign a **User** a new password when theirs is forgotten. |
| Basic Path | 1. The **User** presses the "Forgot Password" button from the Main Menu.<br>2. **RECDAWGS** brings the **User** to a password recovery menu.<br>3. The **User** inputs their email address to have a new password sent to them.<br>4. The **User** presses the "Send New Password" button.<br>5. The **User** goes to their email account and follows the provided link.<br>6. The **User** is presented with two text boxes, New Password and Retype New Password.<br>    **a.** **User** fills these two text boxes out with new password.<br>7. The **User** presses the "Save Password" button and is prompted to log back in with the new password. |
| Alternative Paths | 1. The **User** presses the cancel button, the password edit ceases, and the Use Case is terminated. |
| Exception Paths | 1. If the **User** inputs an email address that does not exist in the email database in step 3 then they will be notified that the email is not in **RECDAWGS** and reprompted for an email adress in step 3. |
| Extension Points | |
| Triggers | The **User** presses the "Forgot Password" button. |
| Assumption | The **User** can not log into **RECDAWGS**. |
| Pre-conditions | The **User** is not logged into **RECDAWGS**. |
| Post-conditions | The **User** password is edited in the database. |

**3.5.2.2.9 - CreateTeam**

| | |
|---|---|
| Name | Create Team |
| ID | CreateTeam |
| Version | 1.0 |
| Author | Justin Tumale |
| Date | 01/29/2016 |
| Summary | Use case to create a new **Team** that will be stored in a **League**. |
| Basic Path | 1. The **User** selects “Leagues” from the main menu.<br>a. **RECDAWGS** responds by bringing the **User** to the “Leagues” menu.<br>2. From the “Leagues” menu, the **User** selects a **League** and presses “View League Teams”.<br>a. **RECDAWGS** responds by bringing the user to the “League Teams” menu.<br>3. From the “League Teams” menu, the **User** presses “Create New Team”.<br>a. The system brings the **User** to the Create New Team screen.<br>4. The **User** enters a team name, and then presses submit.<br>a. **RECDAWGS** brings the **User** to the My Team screen. |
| Alternative Paths | 1. Pressing the cancel button while in step 4 sends the **User** back to the “League Teams” menu. |
| Exception Paths | 1. If in step 3, the **Player** is already registered to a team, the system displays 'You are already registered for a team. Please leave your current team before creating a new one.' |
| Extension Points | |
| Triggers | The **User** decides the create a new team. |
| Assumption | The **Player** is not registered with a current team. |
| Pre-conditions | The **User** is logged into **RECDAWGS**. |
| Post-conditions | The **Team** is added to the database, and the **User** becomes a **Captain**. |

**3.5.2.2.10 - Join Team**

| | |
|---|---|
| Name | Join Team |
| ID | JoinTeam |
| Version | 1.0 |
| Author | Jerry Springfield Jr. |
| Date | 01/29/2016 |
| Summary | Use case for a **User** to join an existing **Team**. |
| Basic Path | 1. The **User** selects “Leagues” from the main menu.<br>a. **RECDAWGS** responds by bringing the **User** to the “Leagues” menu.<br>2. From the “Leagues” menu, the **User** selects a **League** and presses “View League Teams”.<br>a. **RECDAWGS** responds by bringing the user to the “League Teams” menu.<br>3. From the “League Teams” menu, the **User** presses “Join a Team”.<br>a. The system brings the **User** to the Join a Team screen that has a list of the existing **Teams** in that **League**.<br>4. The **User** presses the “Join Team” button next to the name of an existing **Team** from the list of **Teams**.<br>a. **RECDAWGS** brings the **User** to the My Team screen. |
| Alternative Paths | 1. Pressing the cancel button while in step 4 sends the **User** back to the “League Teams” menu.<br>2. Pressing the “Create New Team” button in step 3 while in the **Leagues** menu would instead go to the Create Team use case. |
| Exception Paths | If in step 3, the **Player** is already registered to a team, the system displays 'You are already registered for a team. You can only join a team once.' |
| Extension Points | |
| Triggers | The **User** decides to join a team. |
| Assumption | The **Player** is not registered with a current team. |
| Pre-conditions | The **User** is logged into **RECDAWGS**. |
| Post-conditions | The **User** is added to the **Team** and database. |

**3.5.2.2.11 - CancelLeague**

| | |
|---|---|
| Name | Cancel League |
| ID | CancelLeague |
| Version | 1.0 |
| Author | Jerry Springfield Jr. |
| Date | 01/29/2016 |
| Summary | Use case for an **Admin** to cancel and remove a **League**. |
| Basic Path | 1. The **Admin** selects “Leagues” from the main menu.<br>a. **RECDAWGS** responds by bringing the **Admin** to the “Leagues” menu.<br>2. From the “Leagues” menu, the **Admin** presses “Cancel a League”.<br>a. **RECDAWGS** responds by bringing the **Admin** to a list of the **Leagues**.<br>3. Next to the **League** that the **Admin** wants to cancel, the **Admin** will press the “Cancel League” button.<br>a. **RECDAWGS** responds by updating the database.<br>4. Emails are sent out to all **Users** of the **Teams** in the cancelled **League** to notify them of the cancellation. |
| Alternative Paths | 1. Pressing the cancel button while in step 3 sends the **Admin** back to the “Leagues” menu. |
| Exception Paths | N/A |
| Extension Points | |
| Triggers | The **Admin** decides to cancel a **League**. |
| Assumption | A **League** exists, and the **Admin** would like to cancel that **League**. |
| Pre-conditions | The **Admin** is accessing the **RECDAWGS** database. |
| Post-conditions | The **League** is removed from the **RECDAWGS** database. |

**3.5.2.2.12 - CreateLeagueSchedule**

| | |
|---|---|
| Name | Create League Schedule |
| ID | CreateLeagueSchedule |
| Version | 1.0 |
| Author | Jerry Springfield Jr. |
| Date | 01/29/2016 |
| Summary | Use case for if the **System** is creating a **League** schedule. |
| Basic Path | 1. The **Admin** selects "**Leagues**" from the main menu.<br>    a. **RECDAWGS** responds by bringing the **Admin** to the "Leagues" menu.<br>2. From the "Leagues" menu, the **Admin** presses "Create League Schedule".<br>3. The **RECDAWGS System** lays out a schedule to assign which team is playing which each week.<br>4. Emails are sent out to all **Users** of the **Teams** in the **League** to notify them of the schedule. |
| Alternative Paths | N/A |
| Exception Paths | 1. If there is already a schedule for the season, the **System** cannot create a new one. |
| Extension Points | |
| Triggers | The season has just begun and the **League** needs a schedule. |
| Assumption | A **League** exists, and they need a schedule for the season. |
| Pre-conditions | The **League** does not have a schedule yet and the season is about to start. |
| Post-conditions | The **League** receives a schedule for all of its teams and is stored in the **RECDAWGS** database. |

**3.5.2.2.13 - ReportMatch**

| | |
|---|---|
| Name | Report Match |
| ID | ReportMatch |
| Version | 1.0 |
| Author | Logan Jahnke |
| Date | 01/31/2016 |
| Summary | Use case for reporting a match to the **System**. |
| Basic Path | 1. The **Admin/Captain** presses the "Report Match" button.<br>    a. **RECDAWGS** responds by presenting a form to the **Admin/Captain**.<br>2. The **Admin/Captain** fills out the form by providing the match score.<br>    a. **RECDAWGS** receives the form, stores the match in the database, and sends confirmation to **Admin** and the **Captain** of both teams. |
| Alternative Paths | 1. The **Admin/Captain** cancels the report and the Use Case is terminated.<br>2. If a **Captain** (A) is reporting a match, and the **Captain** (B) of the other team has not reported the match yet, **Captain** (B) will be notified of a reported match by **RECDAWGS** and will be asked to confirm.<br>    a. Note: If **Captain** (B) does not confirm within three (3) days, the **Admin** will be notified and asked to confirm.<br>    b. After confirmation, step 2a is executed. |
| Exception Paths | 1. If a **Captain** (A) has reported a match, and the **Captain** (B) of the other team reports the match with a different outcome as **Captain** (A) indicated, the **Admin** will be notified and asked to present the final report.<br>    a. After the **Admin** provides the final score, step 2a is executed. |
| Extension Points | |
| Triggers | The **Admin/Captain** presses the “Report Match” button. |
| Assumption | A match has been played. |
| Pre-conditions | The **Captain** is a member of the team that played in the match, and the match has not been reported. |
| Post-conditions | The match is added to the database. |

**3.5.2.2.14 - LeaveTeam**

| | |
|---|---|
| Name | Leave Team |
| ID | LeaveTeam |
| Version | 1.0 |
| Author | Logan Jahnke |
| Date | 01/31/2016 |
| Summary | Use case that allows a **Player** or **Captain** to leave his/her current team. |
| Basic Path | 1. The **Player/Captain** presses the "Leave Team" button.<br>a. **RECDAWGS** responds by presenting a confirmation to the **Player/Captain**.<br>2. The **Player/Captain** confirms the action.<br>a. **RECDAWGS** receives the confirmation, removes the **Player/Captain** from the team database, and sends confirmation to **Player/Captain**. |
| Alternative Paths | 1. The **Player/Captain** cancels the action of leaving the team, and the Use Case is terminated.<br>2. If the **Captain** leaves the team, the **Admin** will be notified and will be required to choose a new captain (see: AppointCaptain). |
| Exception Paths | |
| Extension Points | |
| Triggers | The **Player/Captain** presses the “Leave Team” button. |
| Assumption | The **Player/Captain** is on a team. |
| Pre-conditions | The **Player/Captain** is on the team. |
| Post-conditions | The **Player/Captain** is removed from the team. |

**3.5.2.2.15 - AppointCaptain**

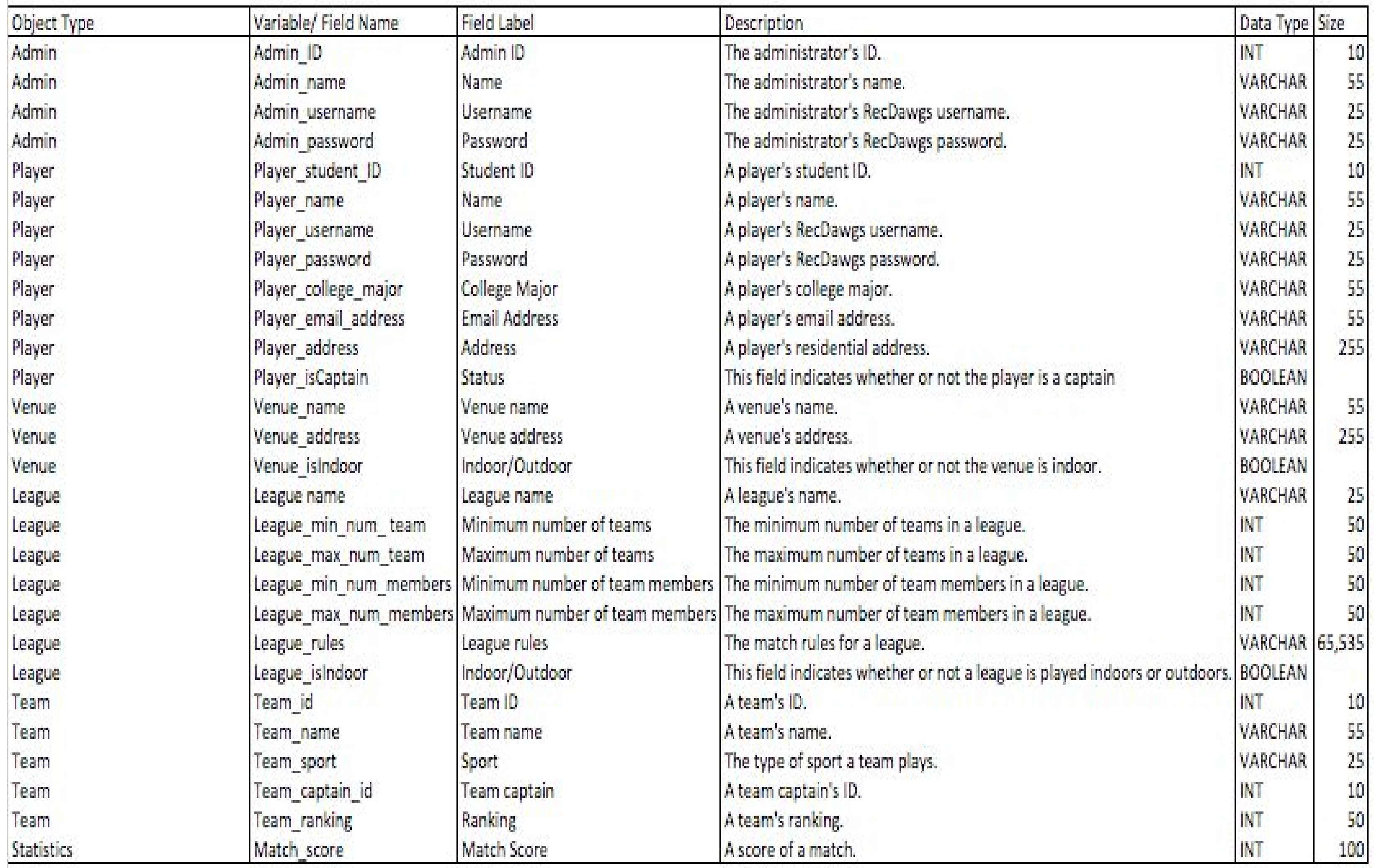
| | |
|---|---|
| Name | AppointCaptain |
| ID | AppointCaptain |
| Version | 1.0 |
| Author | Justin Tumale, Logan Jahnke |
| Date | 01/31/2016 |
| Summary | |
| Basic Path | 1. The Admin presses the "Appoint Captain" button on the Edit Team menu.<br>a. RECDAWGS responds by presenting list of players.<br>2. The Admin selects the captain from the list of players and presses the "Submit" button.<br>a. RECDAWGS receives form, makes changes to the database, and sends confirmation to Admin/Captain.<br>b. The old captain is made a normal player. |
| Alternative Paths | |
| Exception Paths | |
| Extension Points | |
| Triggers | Initiated by the Admin |
| Assumption | |
| Pre-conditions | The Admin is logged into RECDAWGS. |
| Post-conditions | A user is granted Team Captain privileges. |

**3.5.2.2.16 - ViewLeagueStats**

| | |
|---|---|
| Name | View League Statistics |
| ID | ViewLeagueStats |
| Version | 1.0 |
| Author | Logan Jahnke, Bowen Yang |
| Date | 01/31/2016 |
| Summary | Use case that allows a **User** to view a **League's** statistics. |
| Basic Path | 1. The **Admin/Player/Captain** presses the "League Stats" button.<br>a. **RECDAWGS** responds by presenting a page with the **League** statistics to the **User**. |
| Alternative Paths | |
| Exception Paths | 1. If league has yet to start the **Admin/Player/Captain** will be displayed with a message "League has not started" |
| Extension Points | |
| Triggers | The **Admin/Player/Captain** presses the "League Statistics" button. |
| Assumption | A **League** exists. |
| Pre-conditions | A **League** exists. |
| Post-conditions | Nothing changes. |

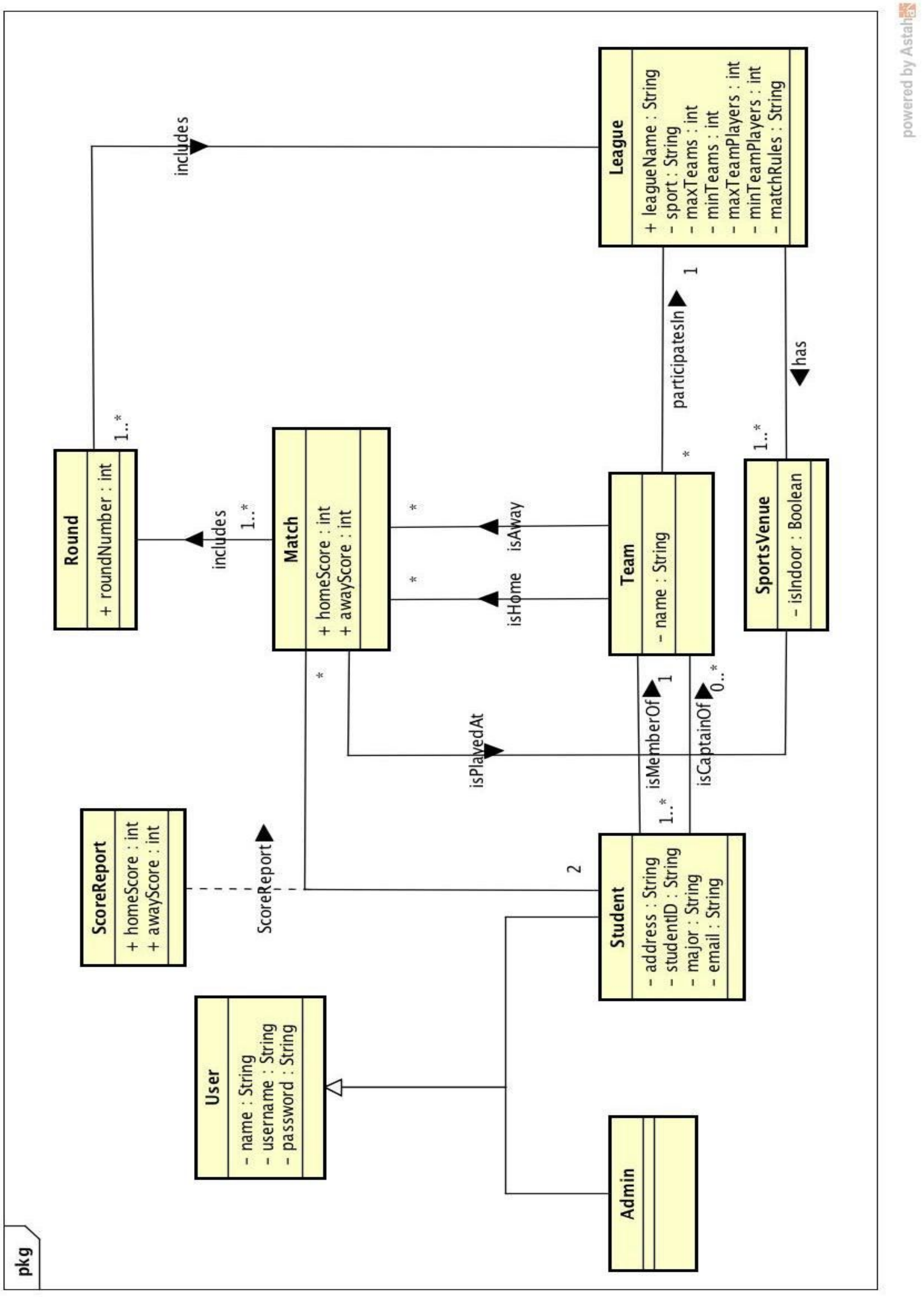
## 3.5.3 - Domain Object Model:

### 3.5.3.1 - Data dictionary:



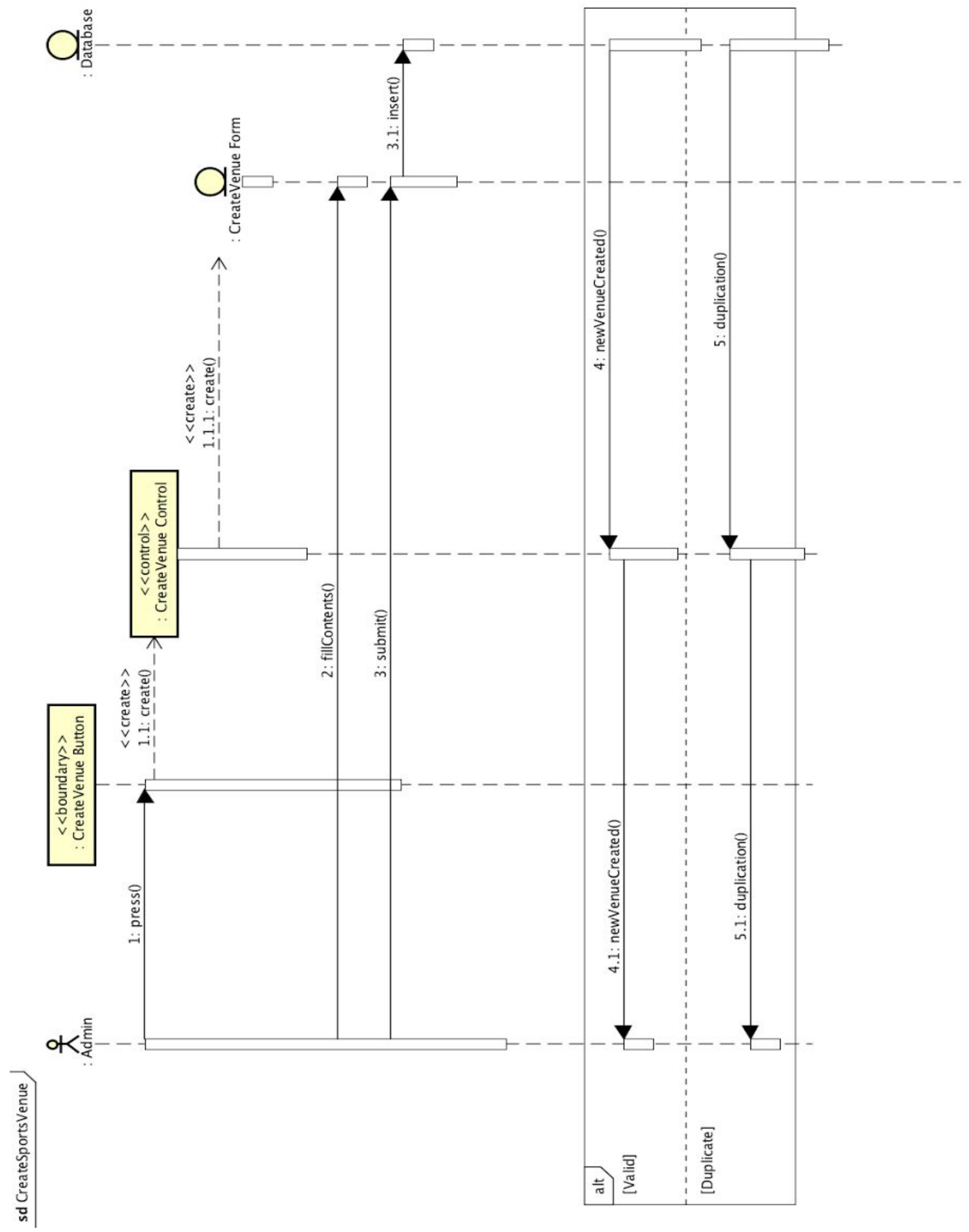
| Object Type | Variable/ Field Name | Field Label | Description | Data Type | Size |
|---|---|---|---|---|---|
| Admin | Admin_ID | Admin ID | The administrator's ID. | INT | 10 |
| Admin | Admin_name | Name | The administrator's name. | VARCHAR | 55 |
| Admin | Admin_username | Username | The administrator's RecDawgs username. | VARCHAR | 25 |
| Admin | Admin_password | Password | The administrator's RecDawgs password. | VARCHAR | 25 |
| Player | Player_student_ID | Student ID | A player's student ID. | INT | 10 |
| Player | Player_name | Name | A player's name. | VARCHAR | 55 |
| Player | Player_username | Username | A player's RecDawgs username. | VARCHAR | 25 |
| Player | Player_password | Password | A player's RecDawgs password. | VARCHAR | 25 |
| Player | Player_college_major | College Major | A player's college major. | VARCHAR | 55 |
| Player | Player_email_address | Email Address | A player's email address. | VARCHAR | 55 |
| Player | Player_address | Address | A player's residential address. | VARCHAR | 255 |
| Player | Player_isCaptain | Status | This field indicates whether or not the player is a captain | BOOLEAN | |
| Venue | Venue_name | Venue name | A venue's name. | VARCHAR | 55 |
| Venue | Venue_address | Venue address | A venue's address. | VARCHAR | 255 |
| Venue | Venue_isIndoor | Indoor/Outdoor | This field indicates whether or not the venue is indoor. | BOOLEAN | |
| League | League name | League name | A league's name. | VARCHAR | 25 |
| League | League_min_num_ team | Minimum number of teams | The minimum number of teams in a league. | INT | 50 |
| League | League_max_num_team | Maximum number of teams | The maximum number of teams in a league. | INT | 50 |
| League | League_min_num_members | Minimum number of team members | The minimum number of team members in a league. | INT | 50 |
| League | League_max_num_members | Maximum number of team members | The maximum number of team members in a league. | INT | 50 |
| League | League_rules | League rules | The match rules for a league. | VARCHAR | 65,535 |
| League | League_isIndoor | Indoor/Outdoor | This field indicates whether or not a league is played indoors or outdoors. | BOOLEAN | |
| Team | Team_id | Team ID | A team's ID. | INT | 10 |
| Team | Team_name | Team name | A team's name. | VARCHAR | 55 |
| Team | Team_sport | Sport | The type of sport a team plays. | VARCHAR | 25 |
| Team | Team_captain_id | Team captain | A team captain's ID. | INT | 10 |
| Team | Team_ranking | Ranking | A team's ranking. | INT | 50 |
| Statistics | Match_score | Match Score | A score of a match. | INT | 100 |

### 3.5.3.2 - Class Diagrams

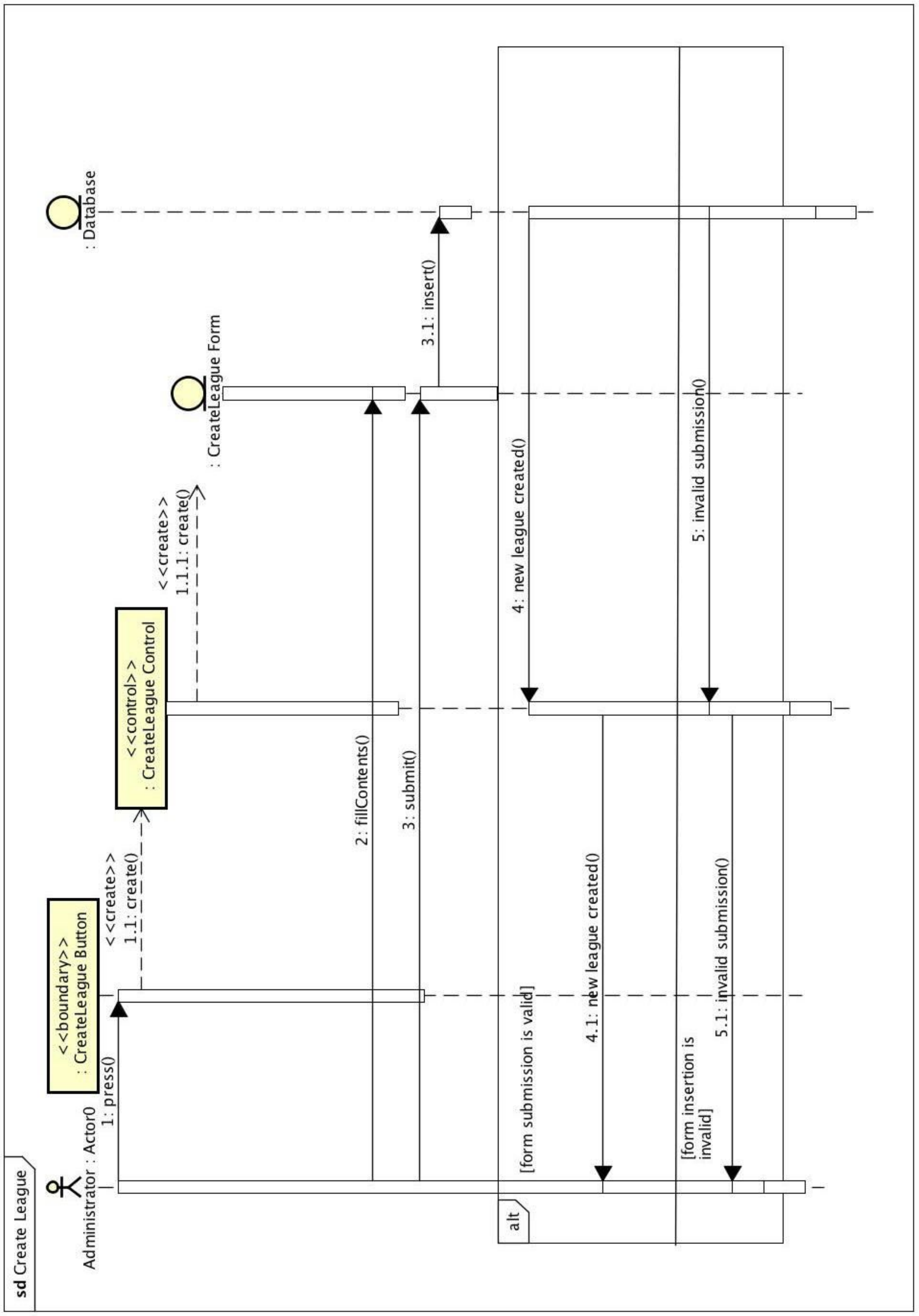


## 3.5.4 - Dynamic Models

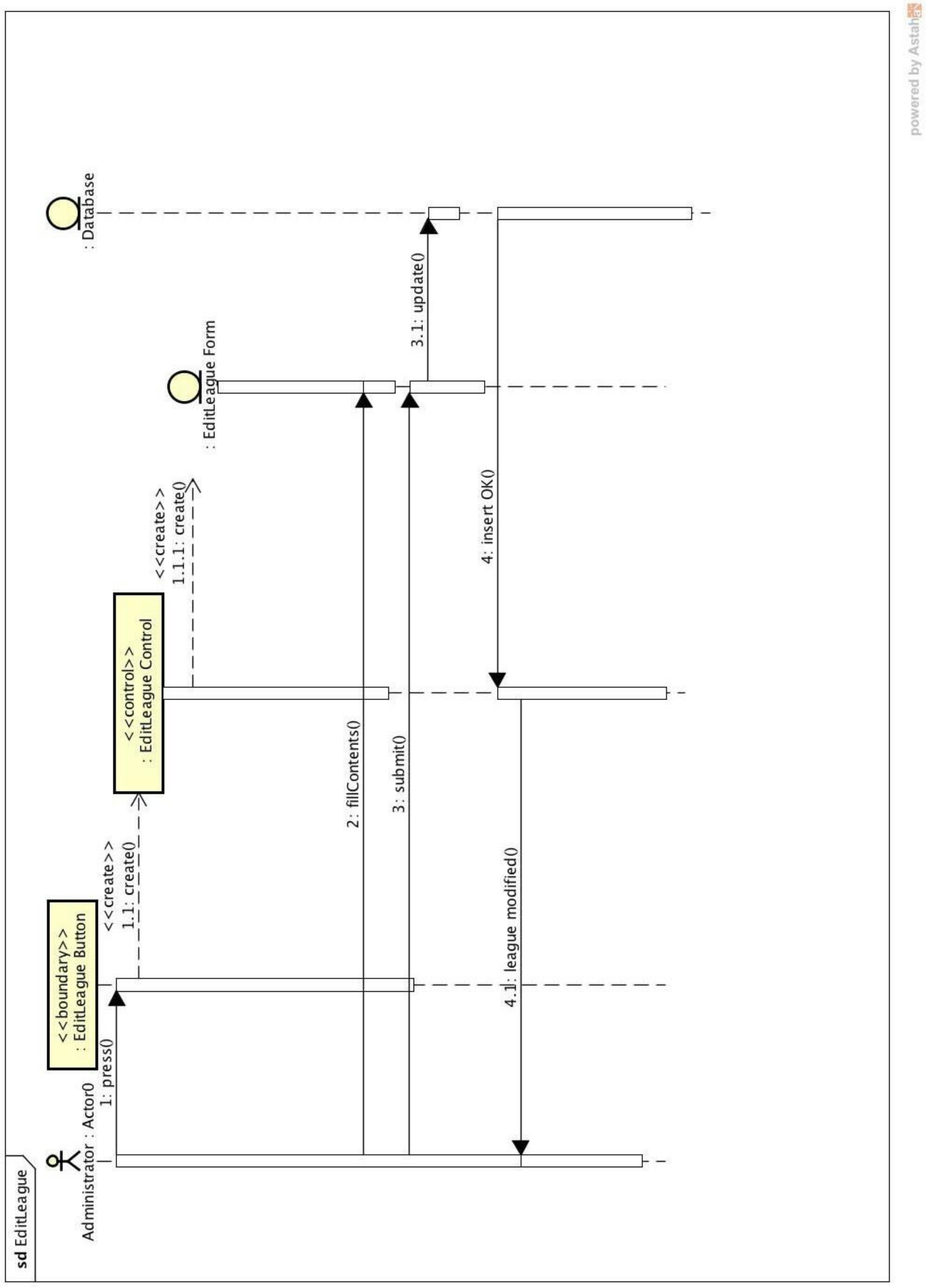
### 3.5.4.1 - CreateSportsVenue



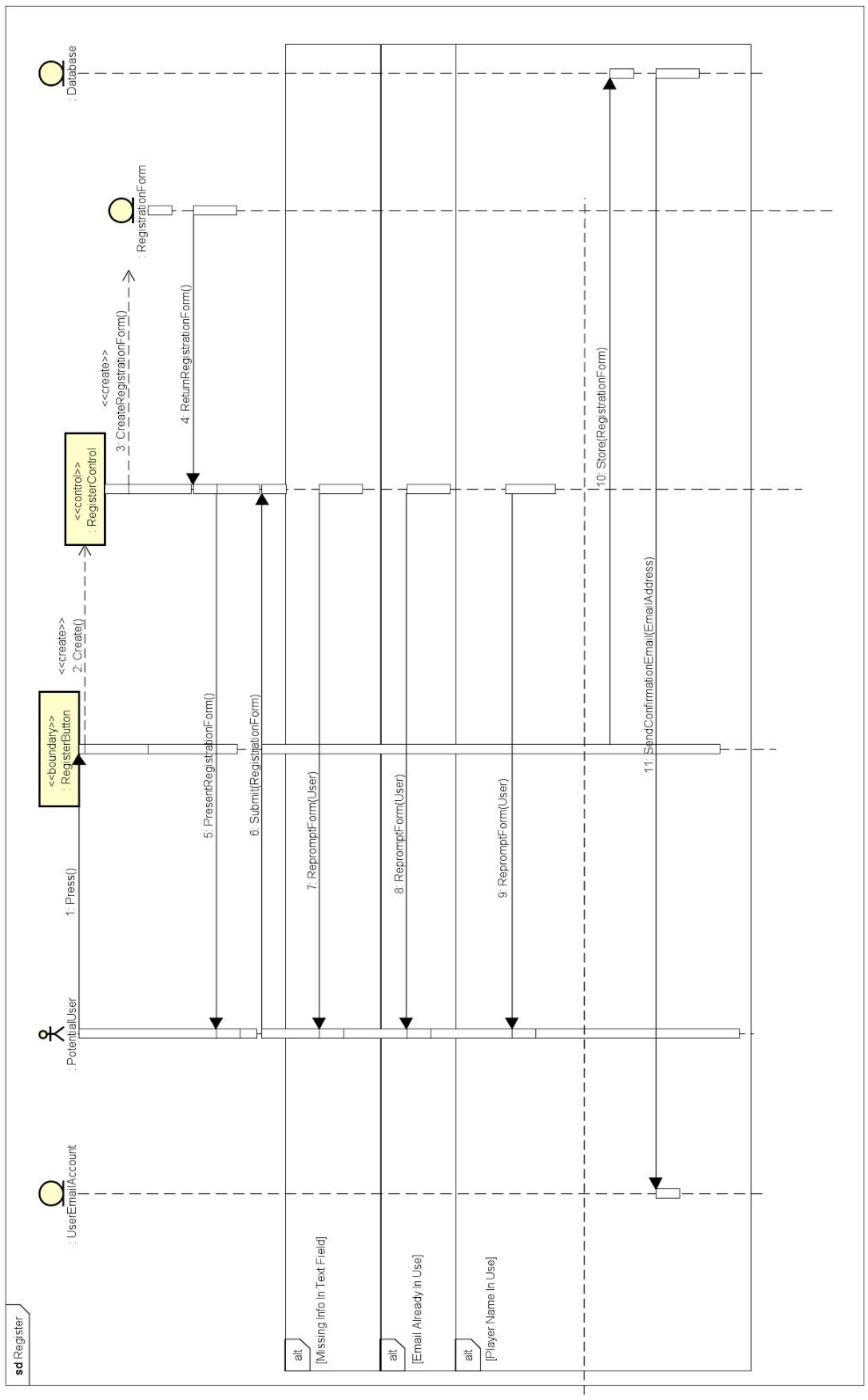
## 3.5.4.2 - CreateLeague



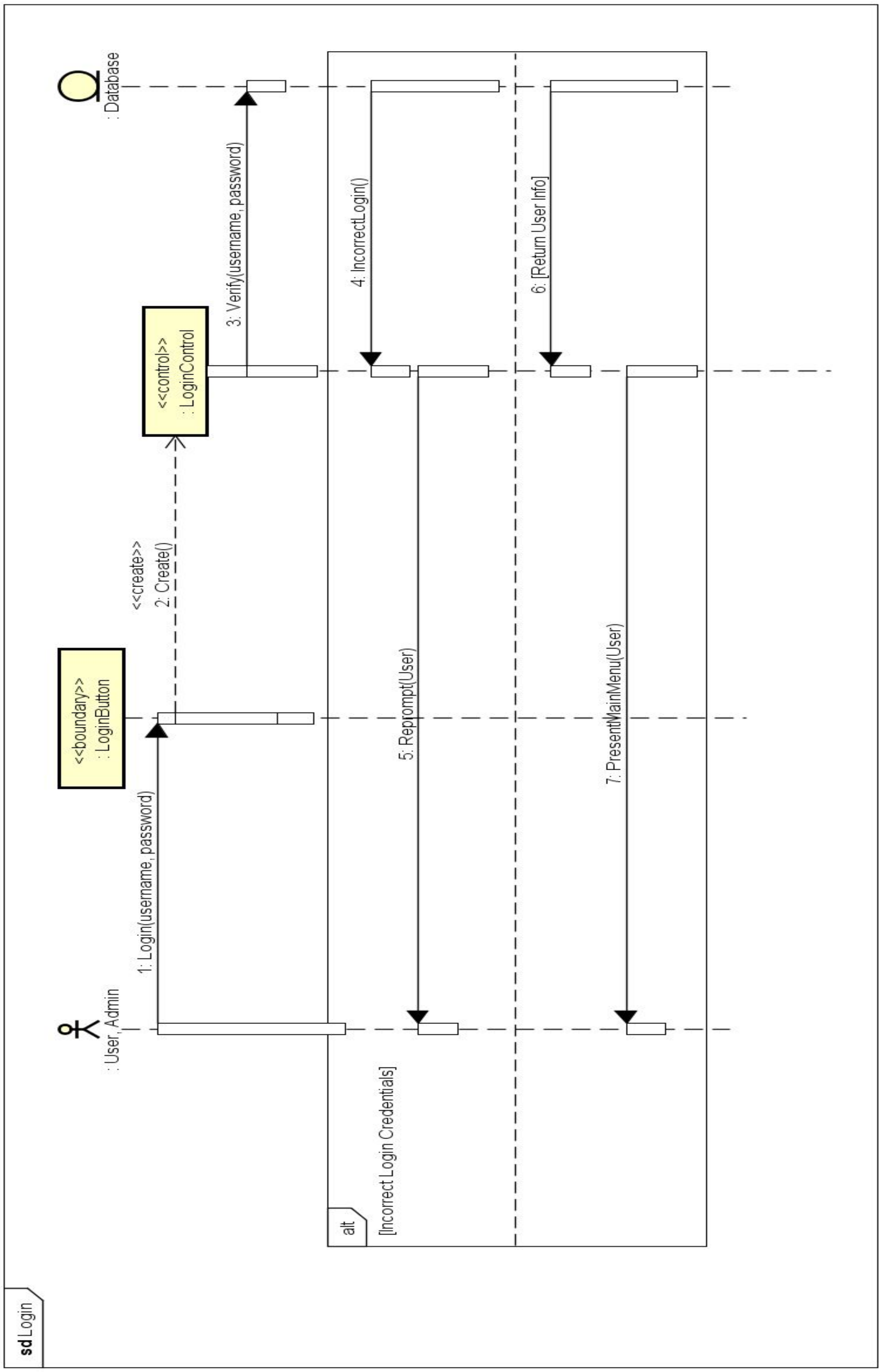
## 3.5.4.3 - EditLeague



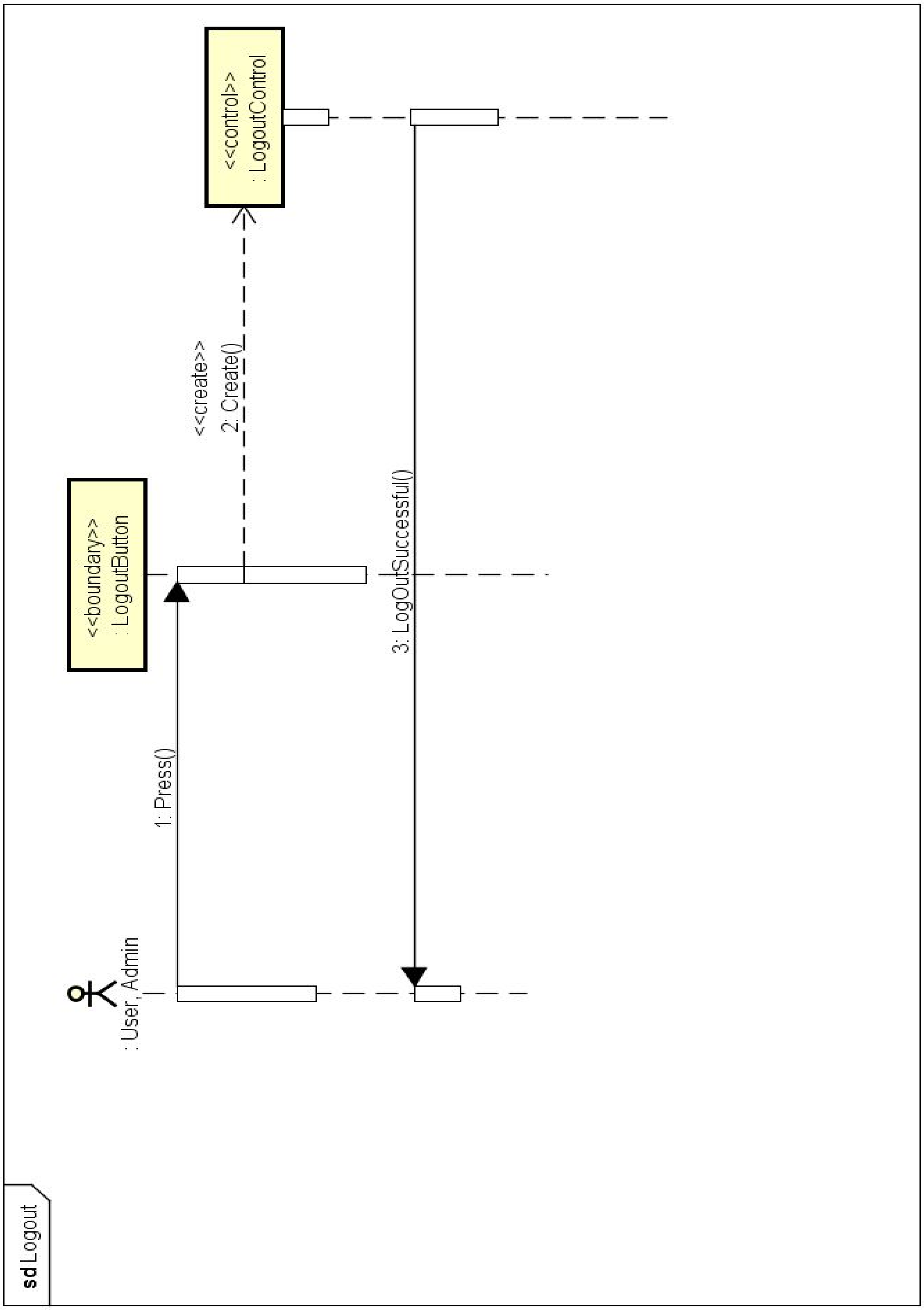
### 3.5.4.4 - Register



### 3.5.4.5 - Login

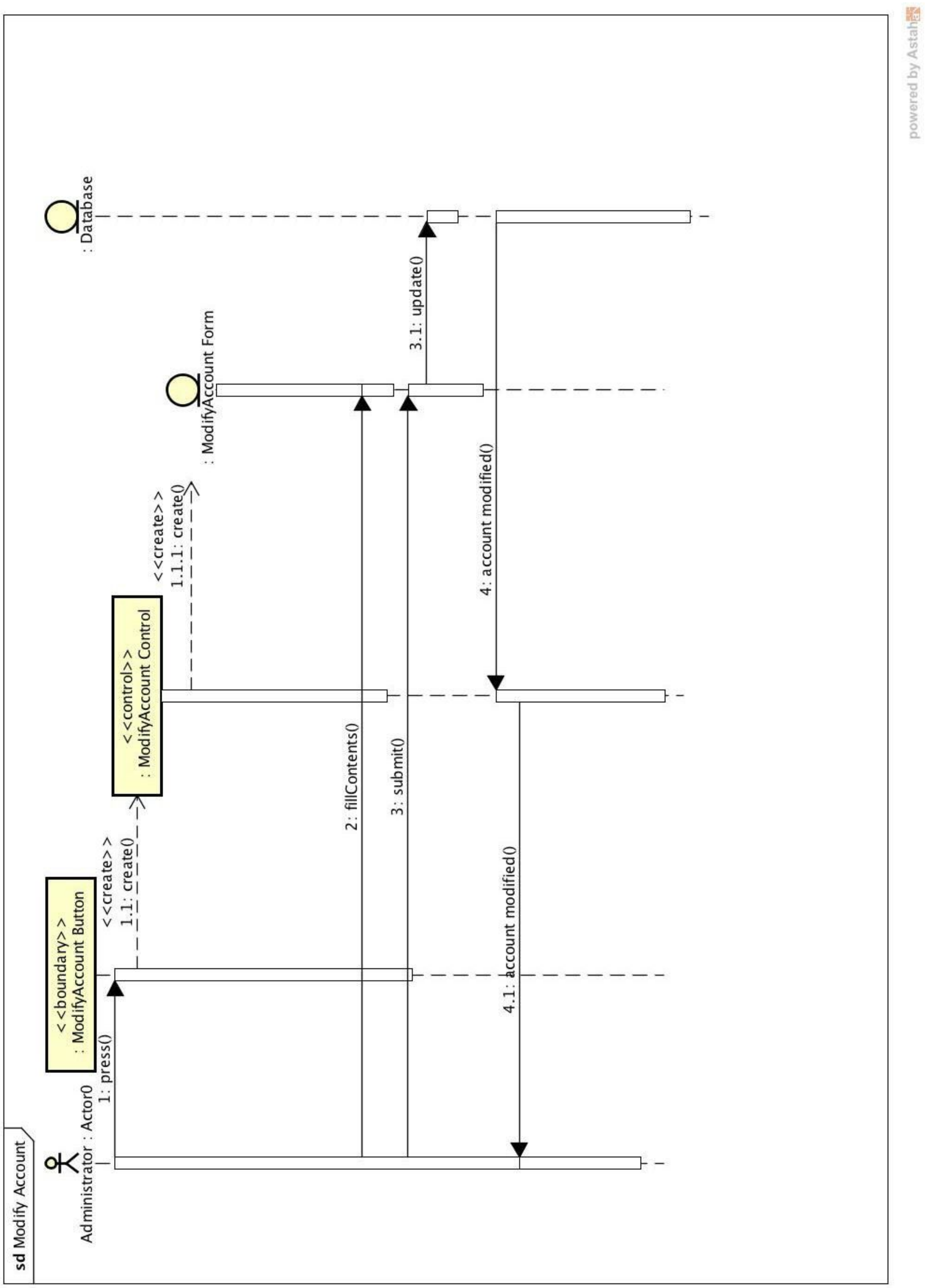

sd Login
: User, Admin
<<boundary>>
: LoginButton
<<control>>
: LoginControl
: Database
1: Login(username, password)
<<create>>
2: Create()
3: Verify(username, password)
alt
[Incorrect Login Credentials]
4: IncorrectLogin()
5: Reprompt(User)
6: [Return User Info]
7: PresentMainMenu(User)

### 3.5.4.6 - Logout

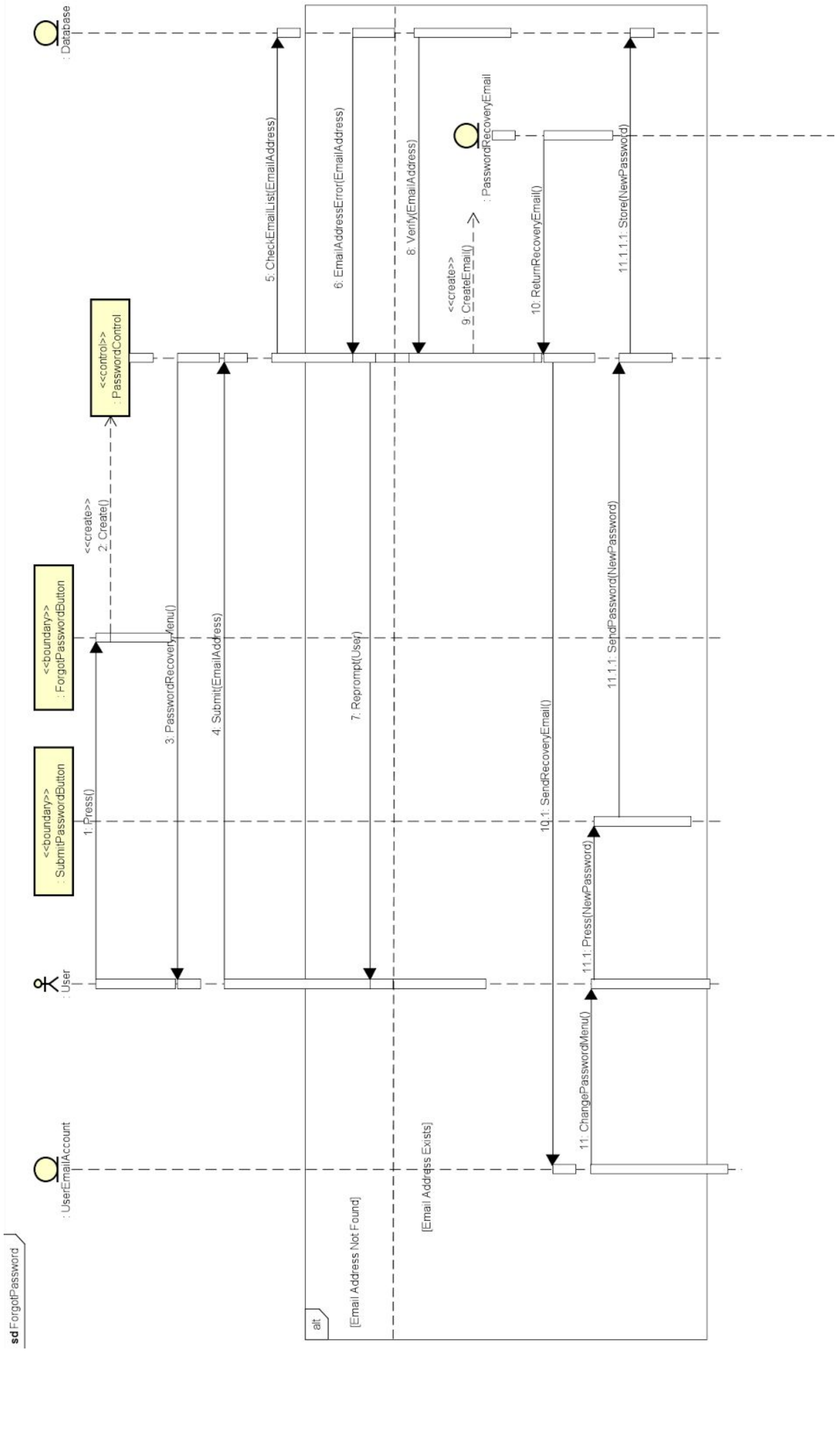


sd Logout
: User, Admin
<<boundary>>
: LogoutButton
<<control>>
: LogoutControl
1: Press()
<<create>>
2: Create()
3: LogOutSuccessful()

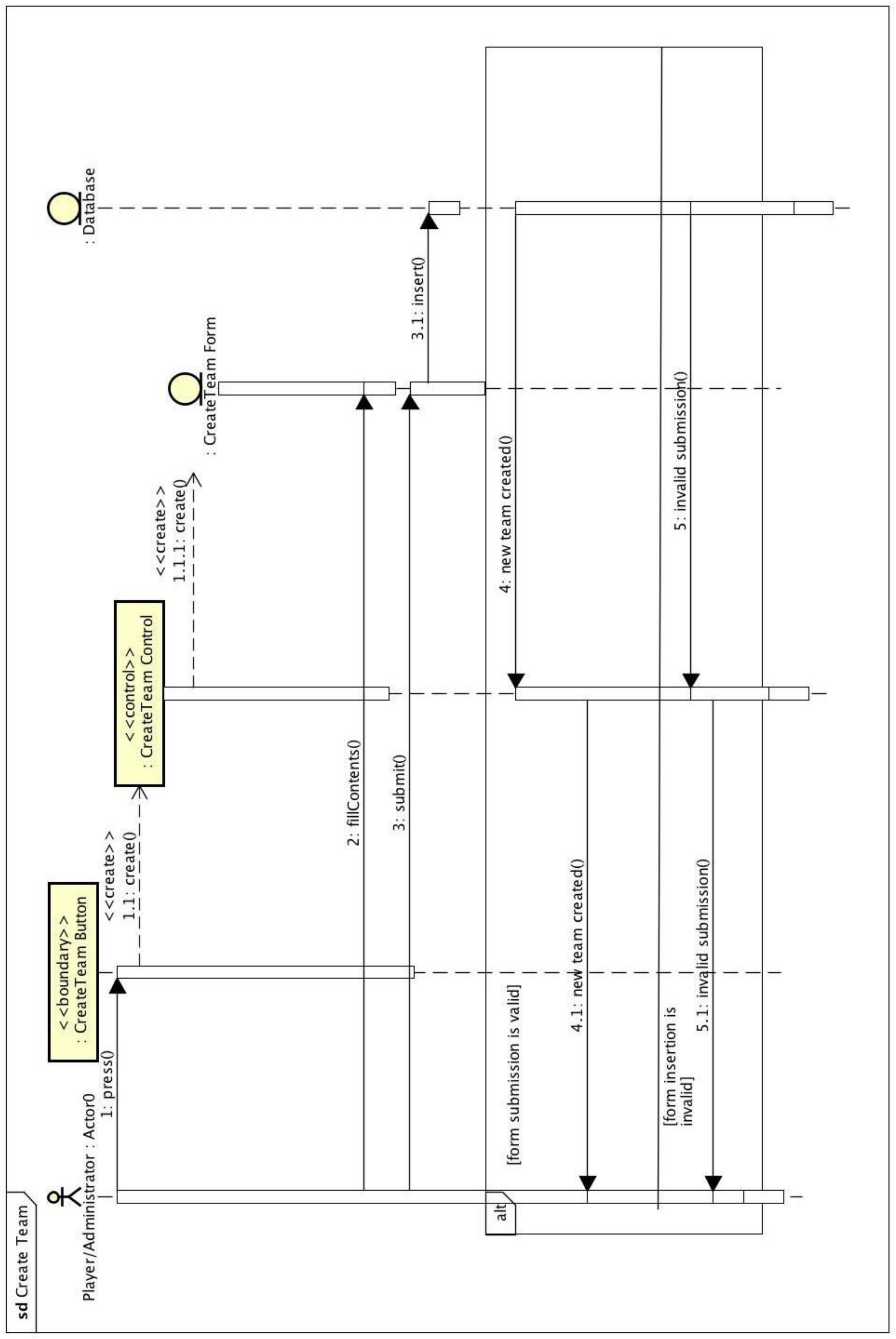
### 3.5.4.7 - Modify Account



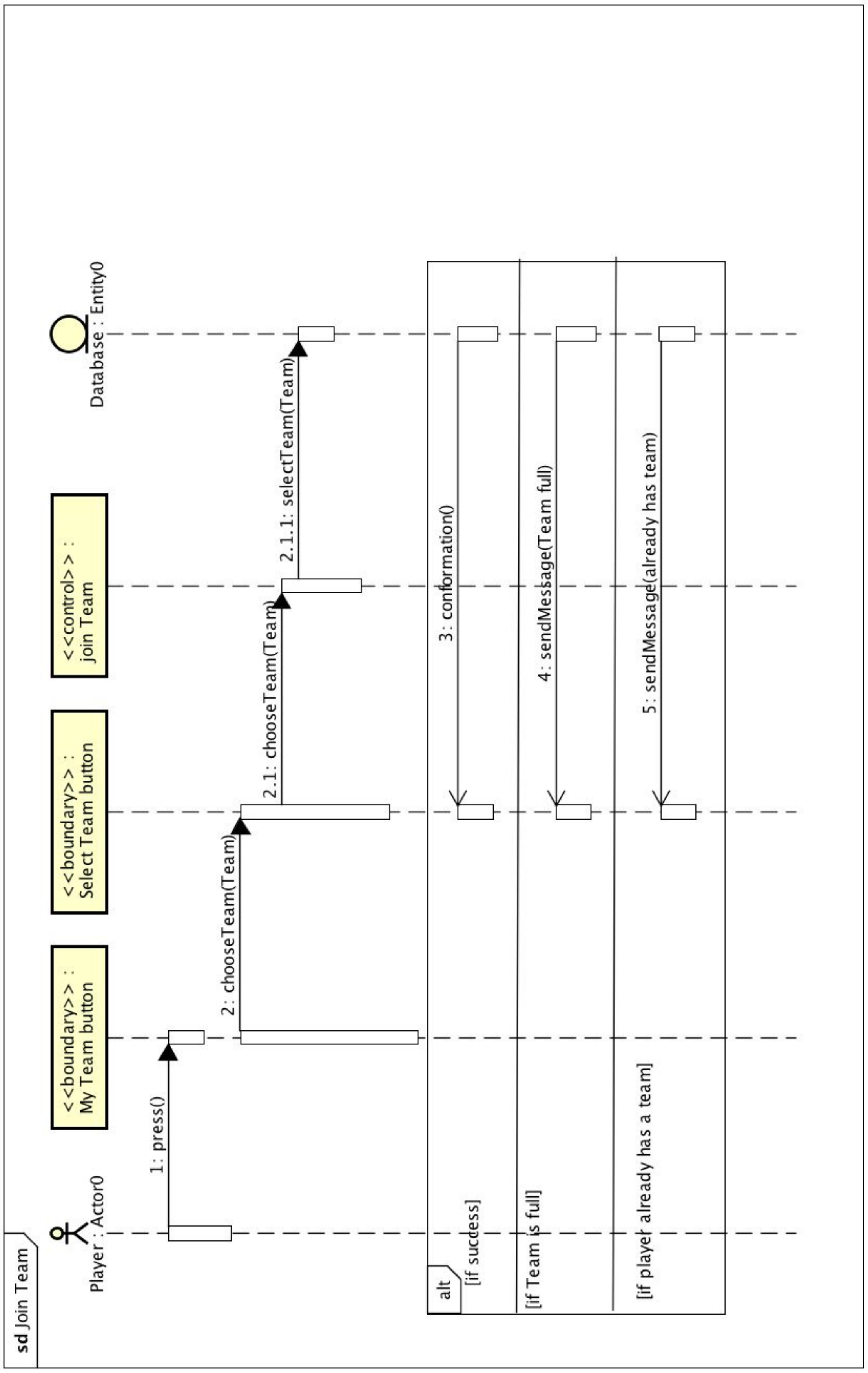
### 3.5.4.8 - ForgotPassword


sd ForgotPassword
: UserEmailAccount
: User
<<boundary>> : SubmitPasswordButton
<<boundary>> : ForgotPasswordButton
<<control>> : PasswordControl
: Database
: PasswordRecoveryEmail
1: Press()
<<create>> 2: Create()
3: PasswordRecoveryMenu()
4: Submit(EmailAddress)
5: CheckEmailList(EmailAddress)
6: EmailAddressError(EmailAddress)
7: Reprompt(User)
8: Verify(EmailAddress)
<<create>> 9: CreateEmail()
10: ReturnRecoveryEmail()
10.1: SendRecoveryEmail()
11: ChangePasswordMenu()
11.1: Press(NewPassword)
11.1.1: SendPassword(NewPassword)
11.1.1.1: Store(NewPassword)
alt
[Email Address Not Found]
[Email Address Exists]

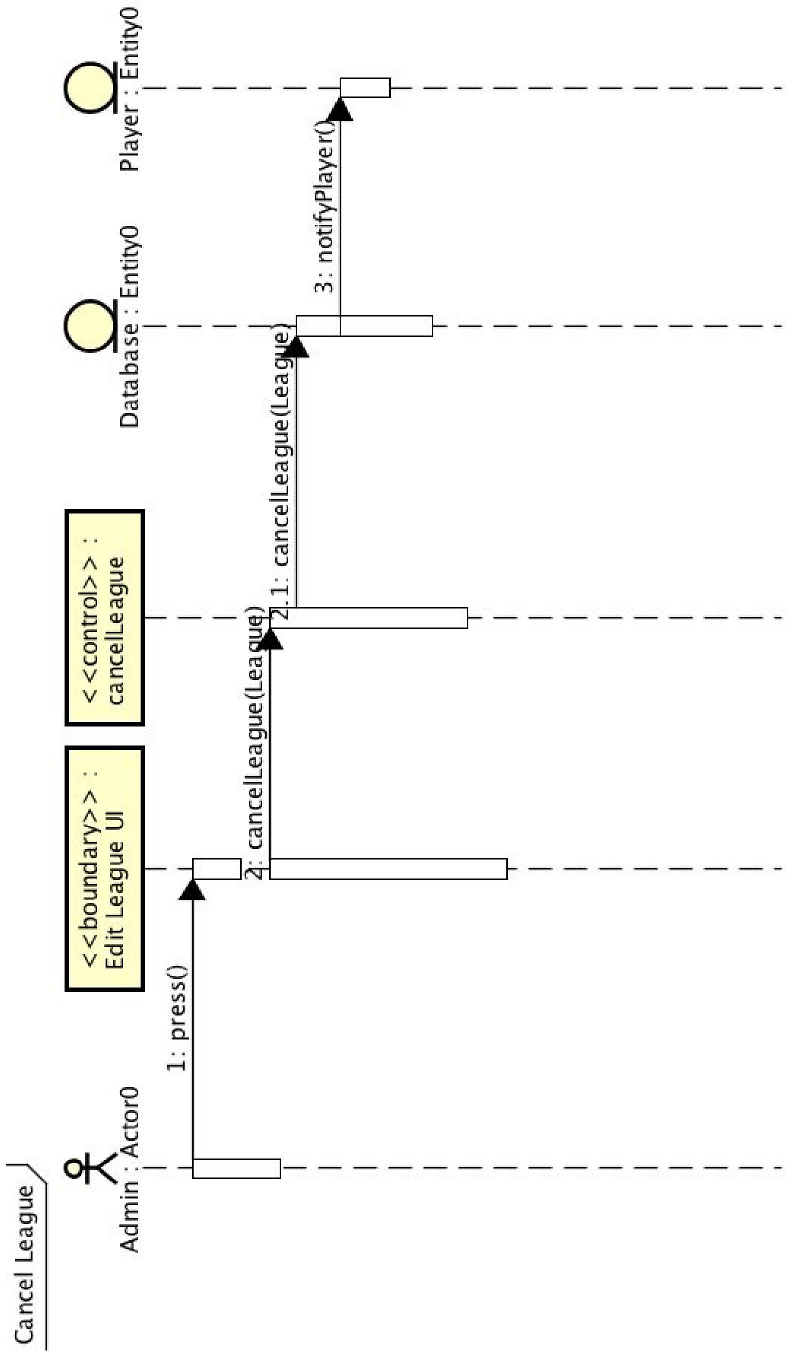
### 3.5.4.9 - CreateTeam



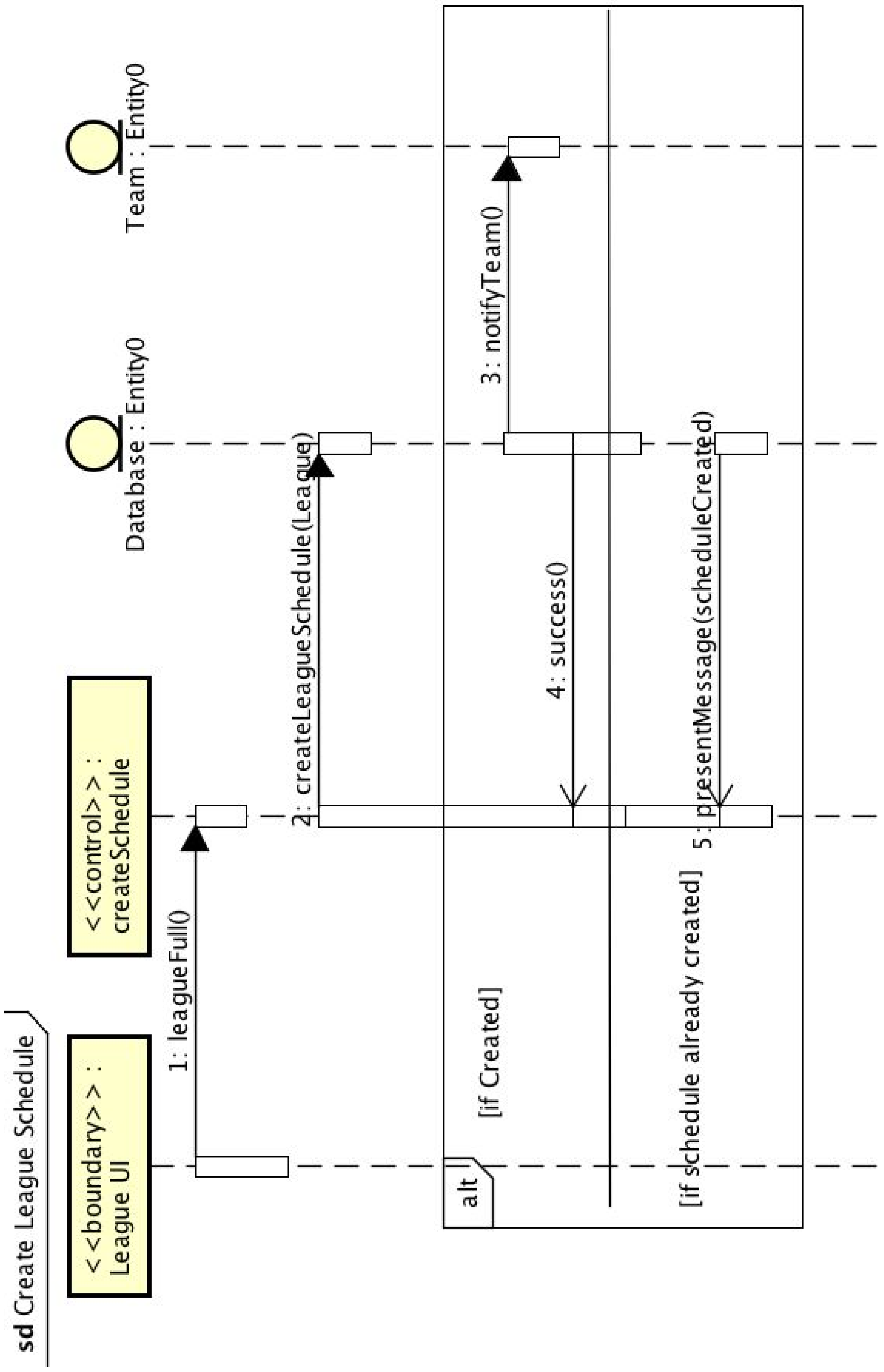
## 3.5.4.10 - JoinTeam



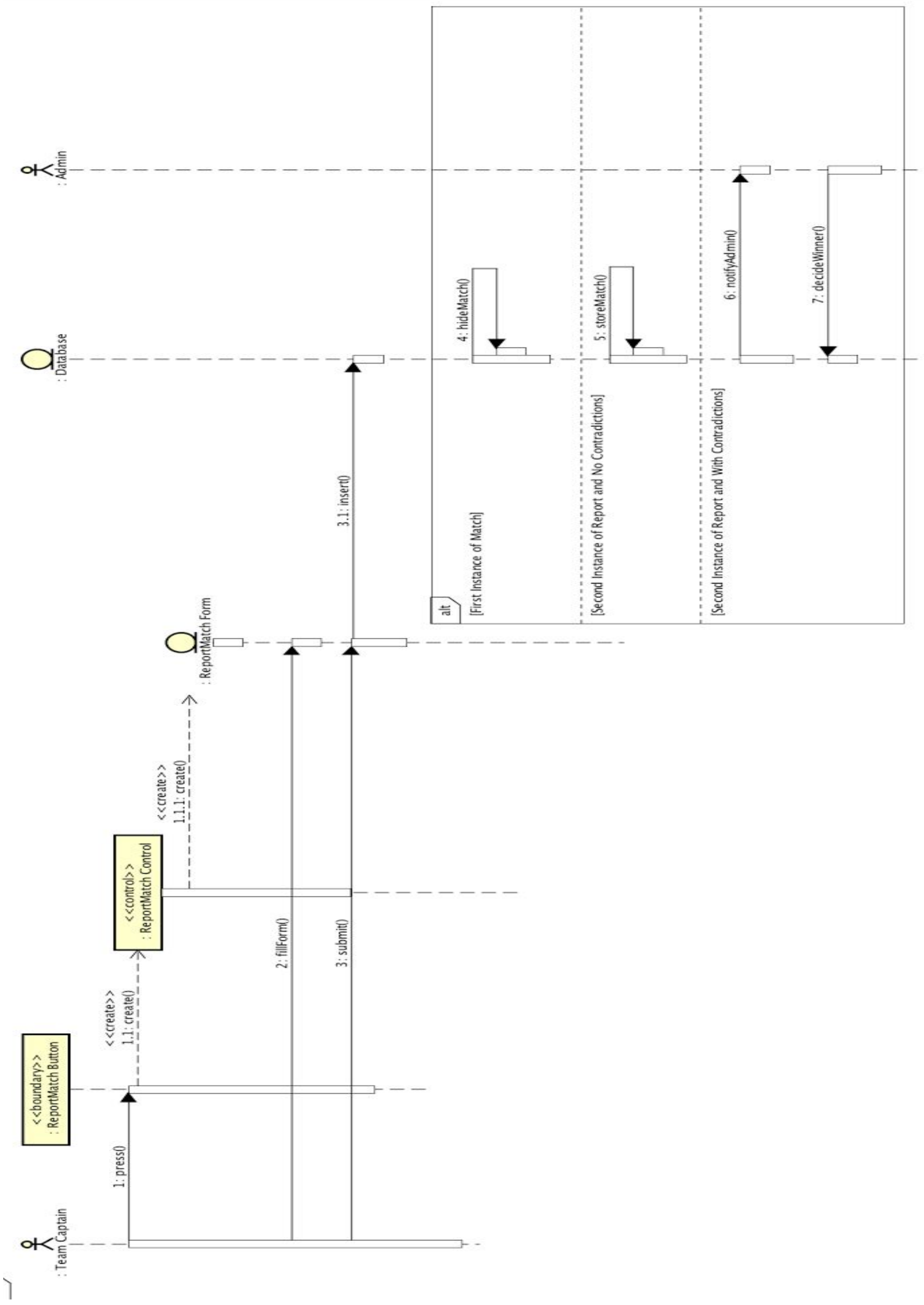
## 3.5.4.11 - CancelLeague



**3.5.4.12 - CreateLeagueSchedule**

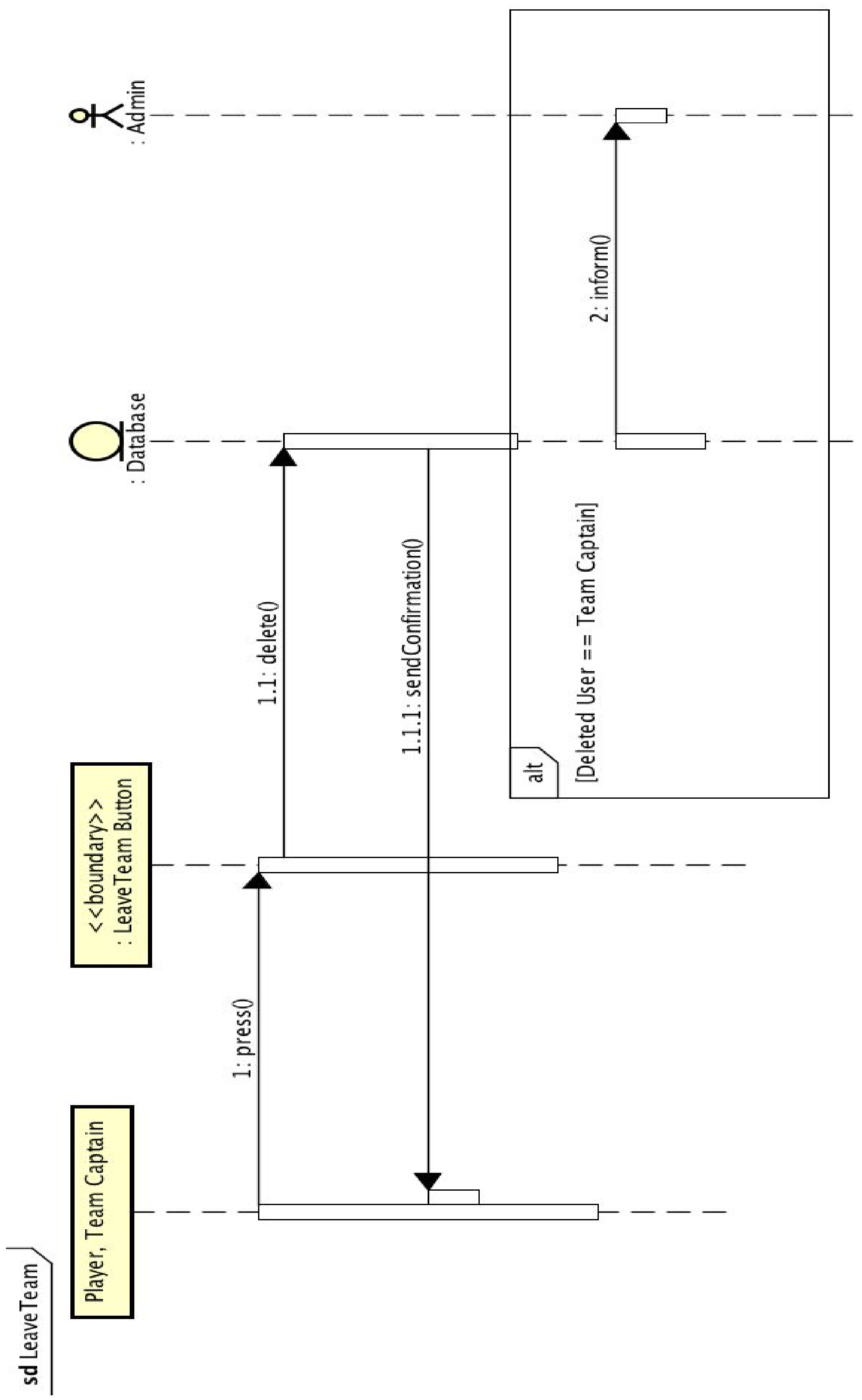

sd Create League Schedule
<<boundary>> : League UI
<<control>> : createSchedule
Database : Entity0
Team : Entity0
1: leagueFull()
2: createLeagueSchedule(League)
3 : notifyTeam()
4 : success()
5 : presentMessage(scheduleCreated)
alt
[if Created]
[if schedule already created]

### 3.5.4.13 - ReportMatch

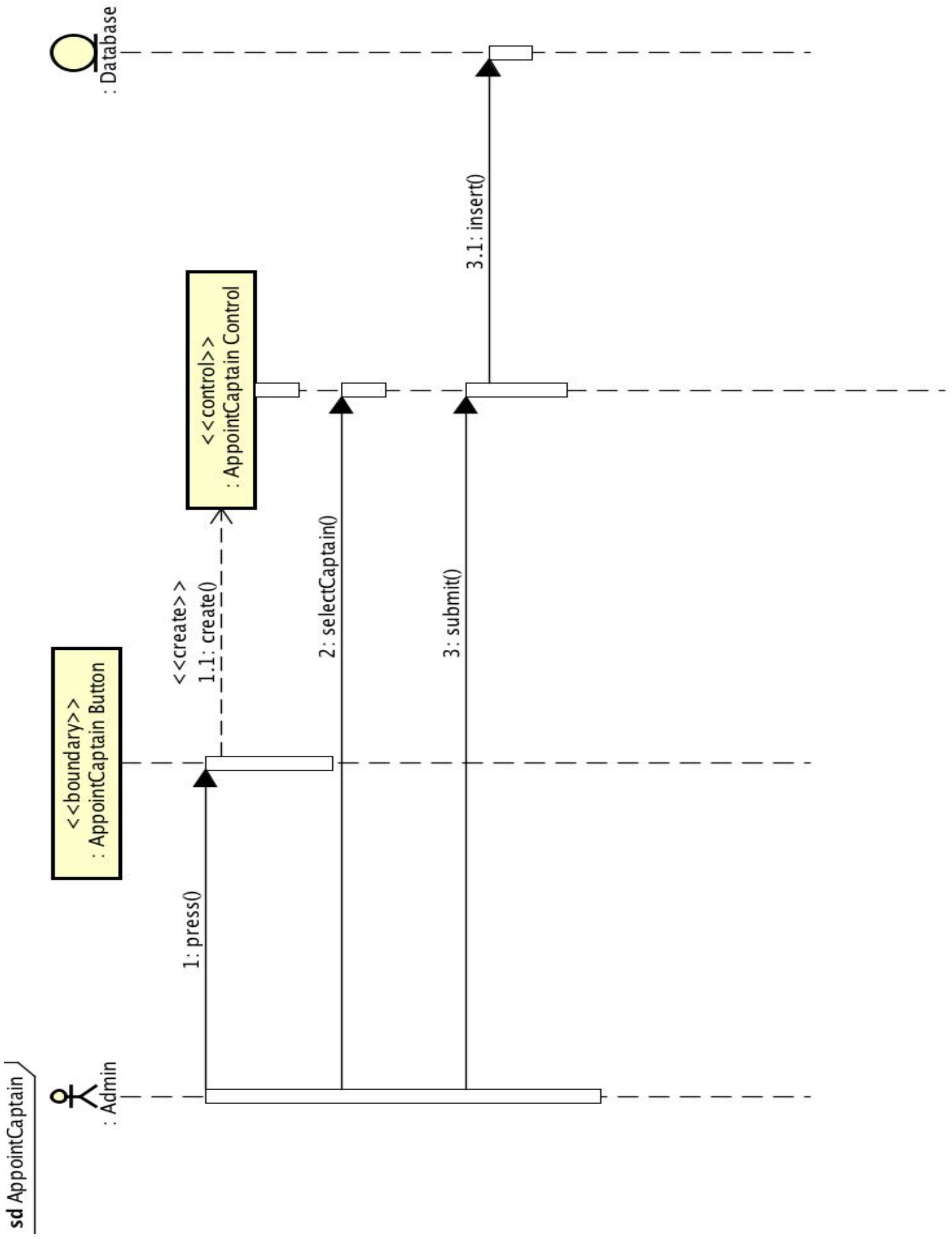


: Team Captain
<<boundary>> : ReportMatch Button
<<control>> : ReportMatch Control
: ReportMatch Form
: Database
: Admin
1: press()
<<create>> 1.1: create()
<<create>> 1.1.1: create()
2: fillForm()
3: submit()
3.1: insert()
alt
[First Instance of Match]
4: hideMatch()
[Second Instance of Report and No Contradictions]
5: storeMatch()
[Second Instance of Report and With Contradictions]
6: notifyAdmin()
7: decideWinner()

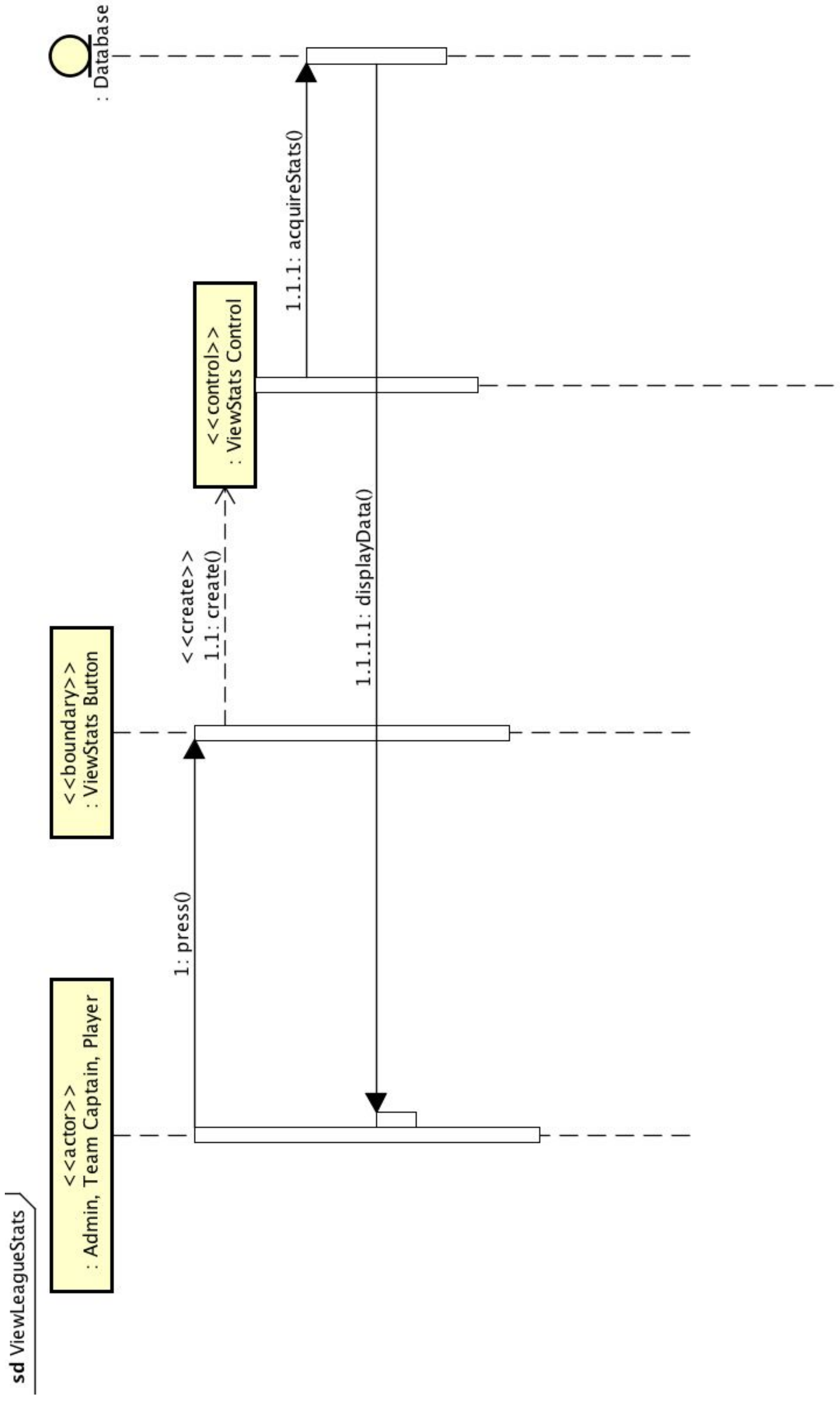
### 3.5.4.14 - LeaveTeam



## 3.5.4.15 - AppointCaptain

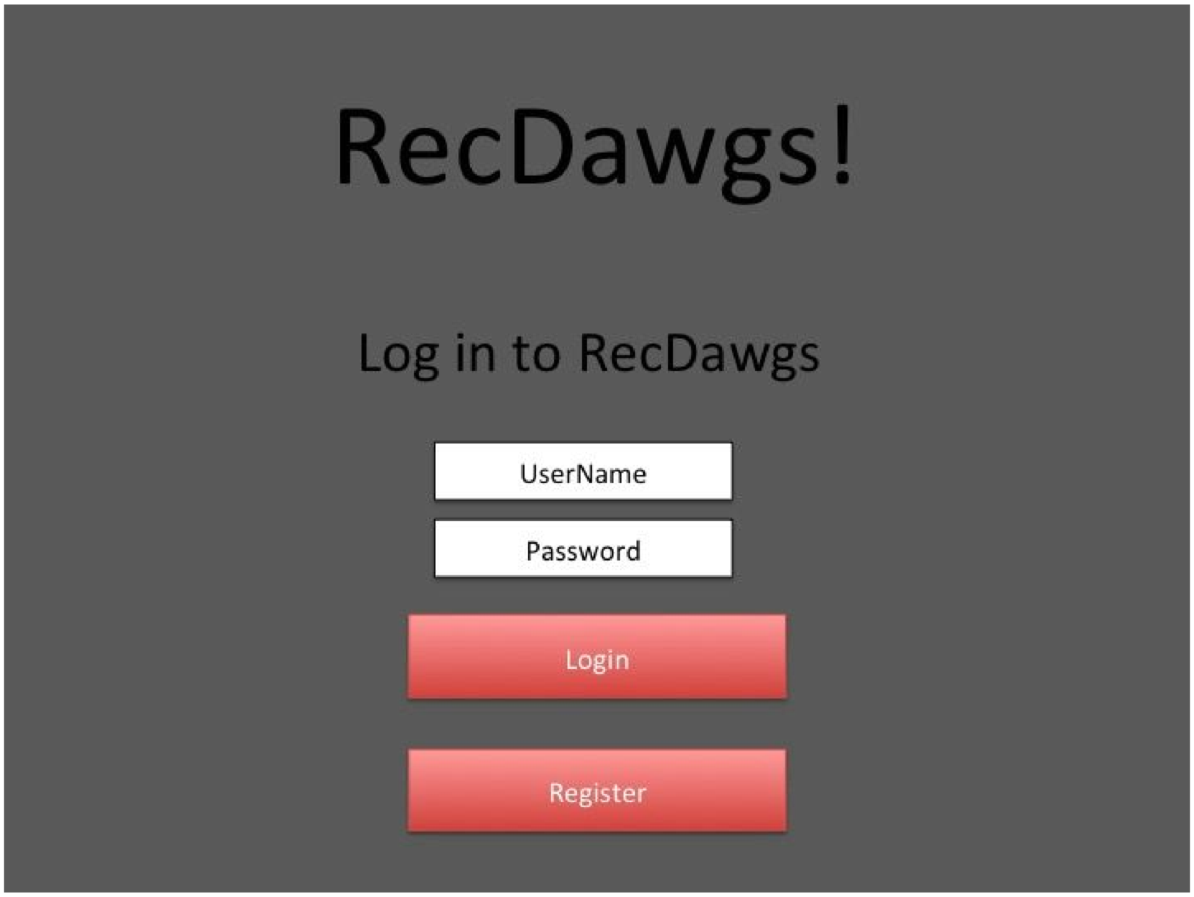


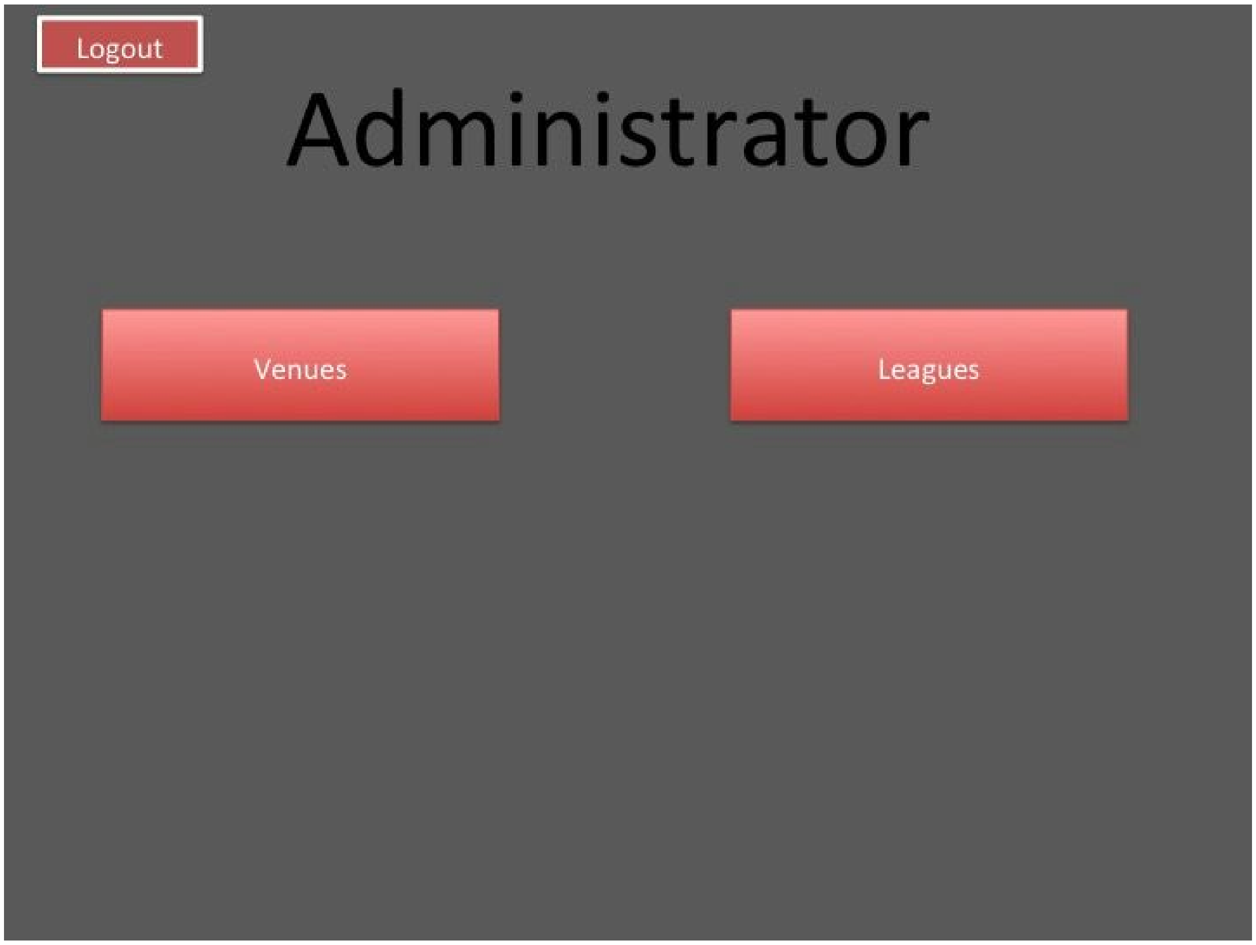
### 3.5.4.16 - ViewLeagueStats

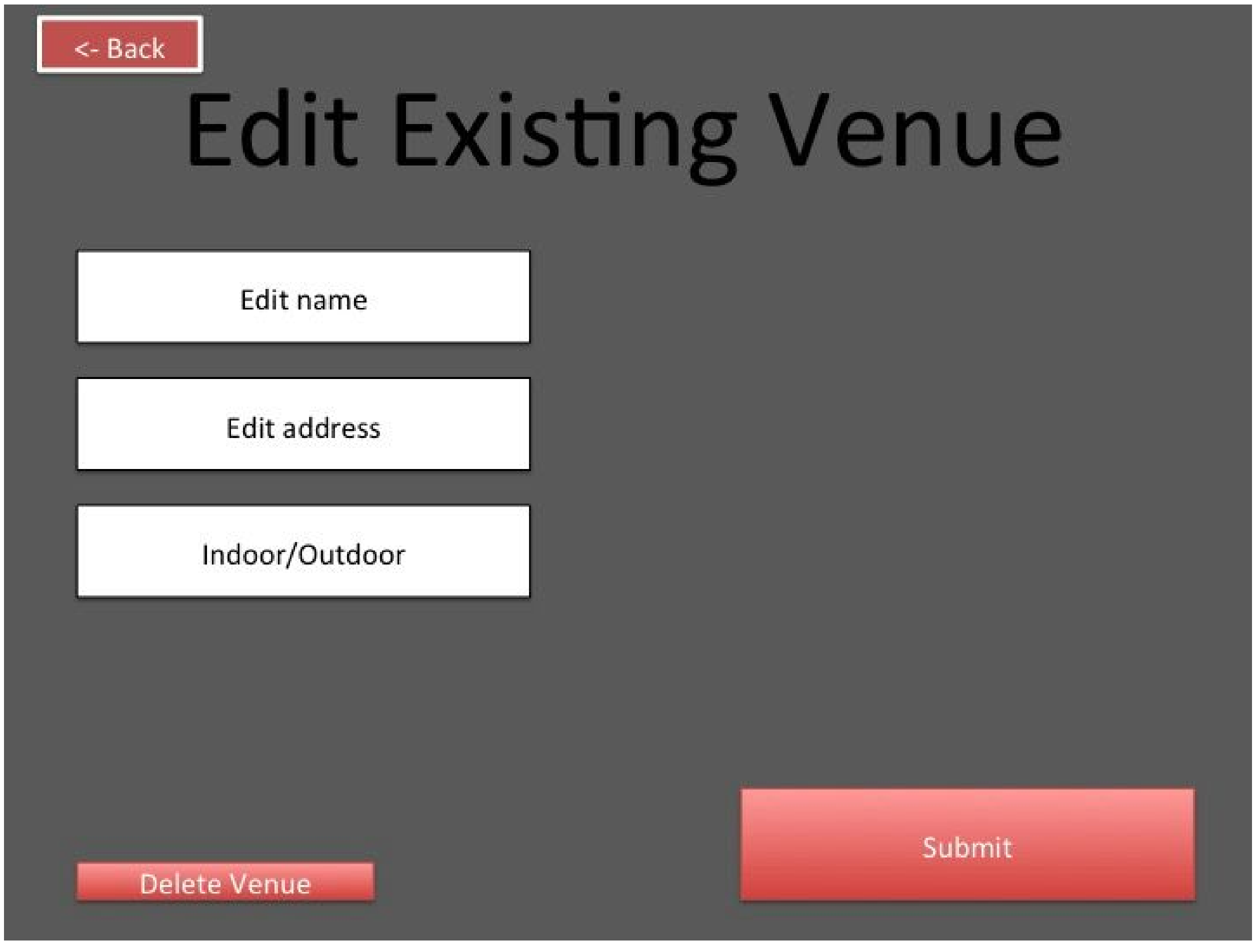


## 3.5.5 - User Interfaces

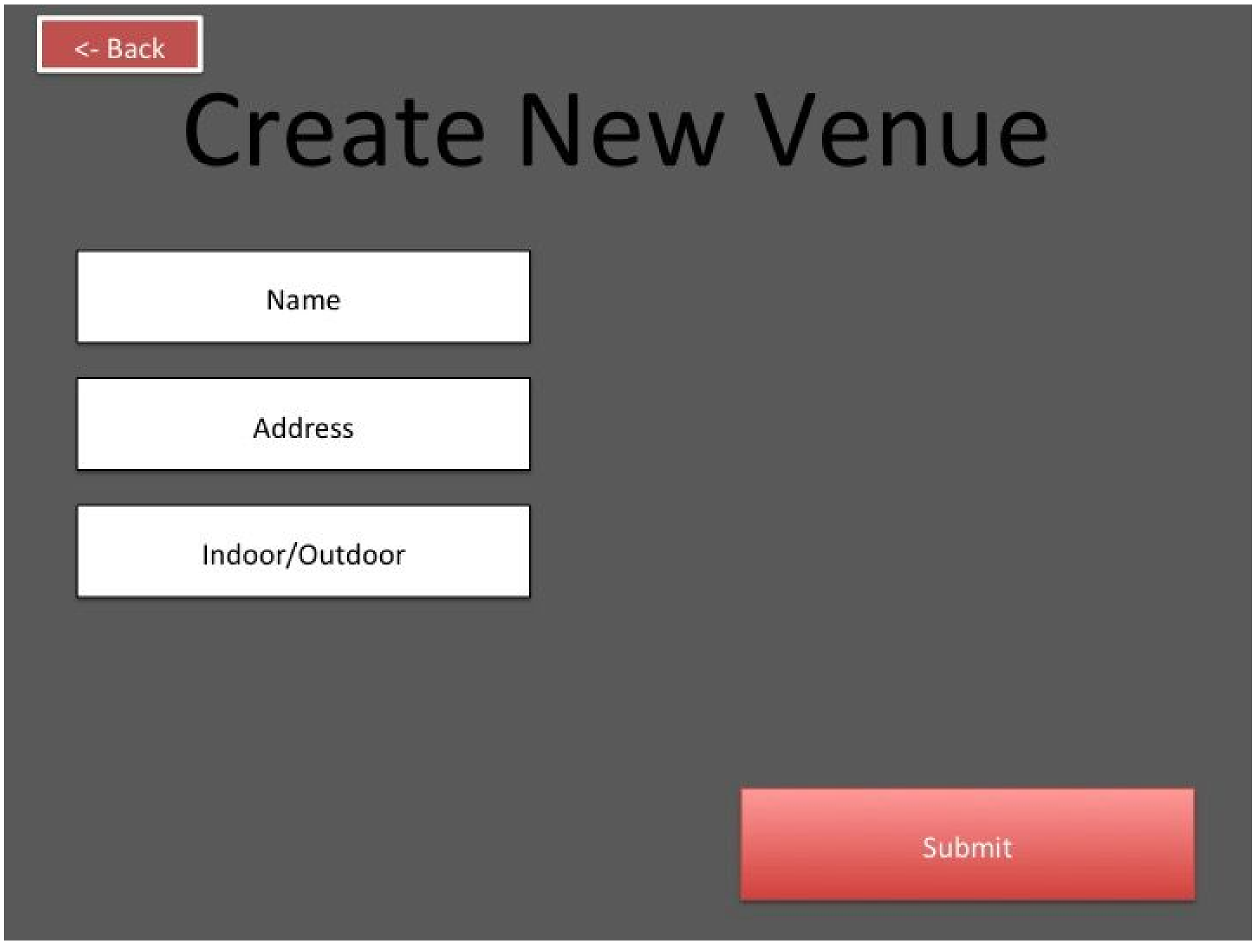
### 3.5.5.1 - User Interfaces: Administrator

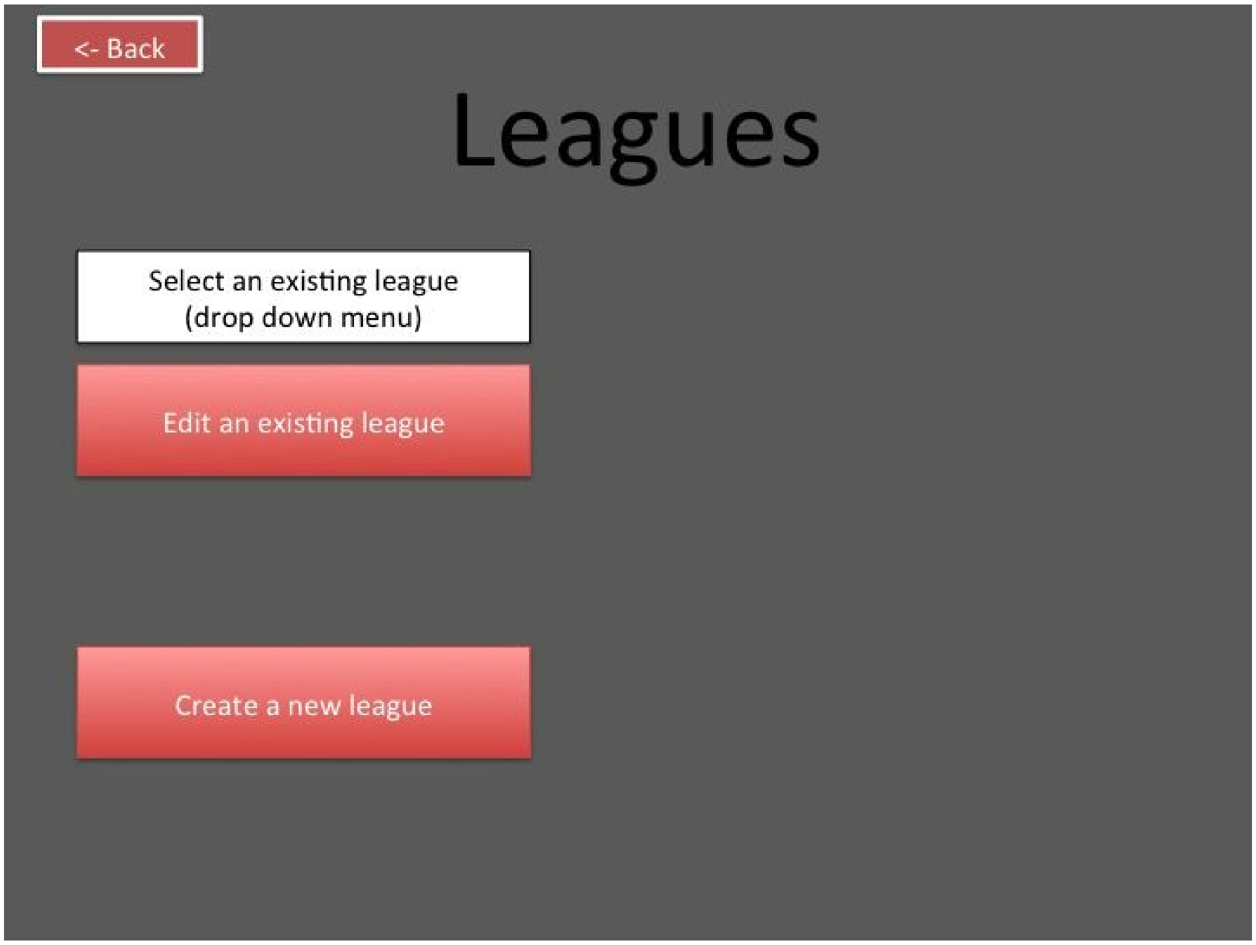


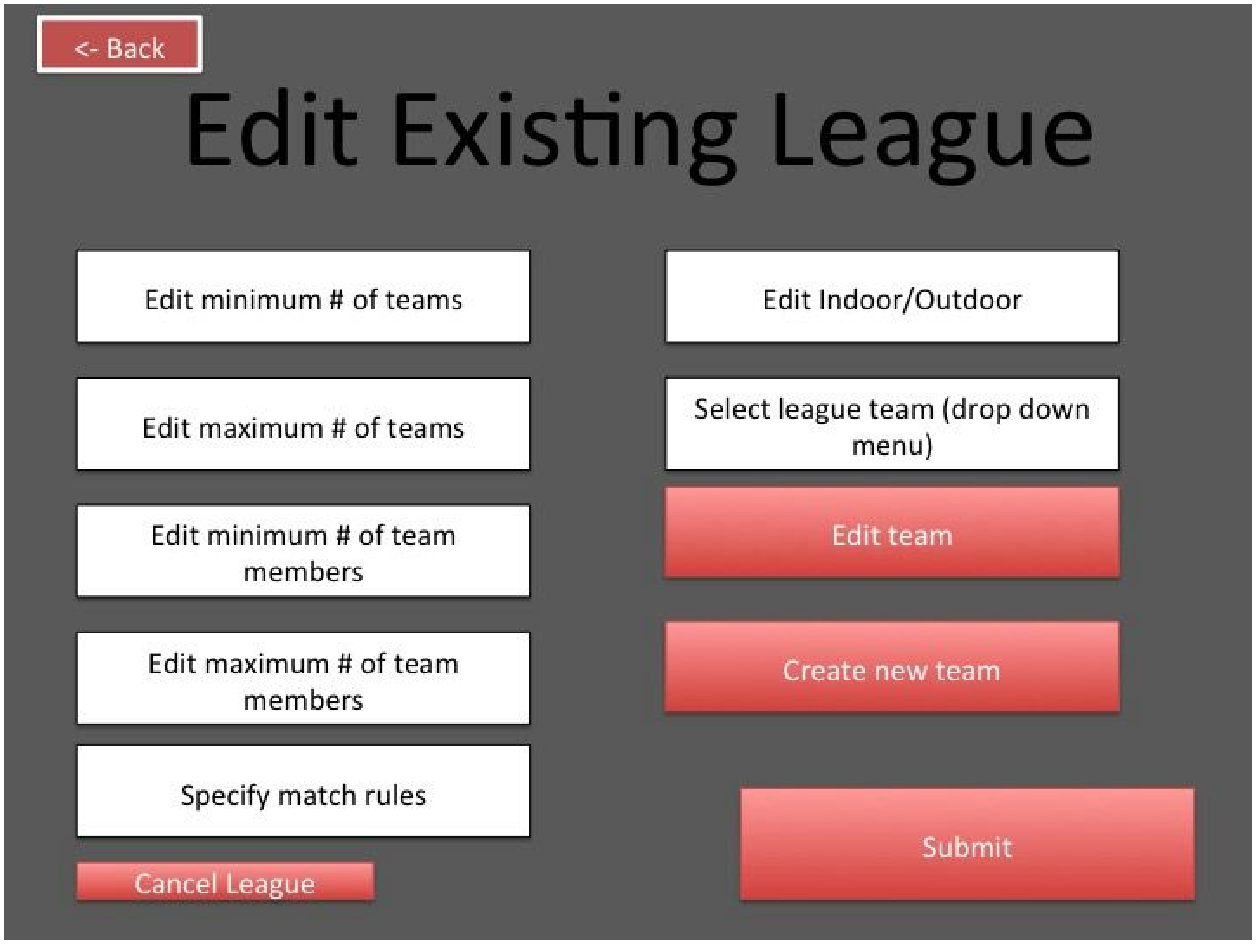

Logout
Administrator
Venues
Leagues



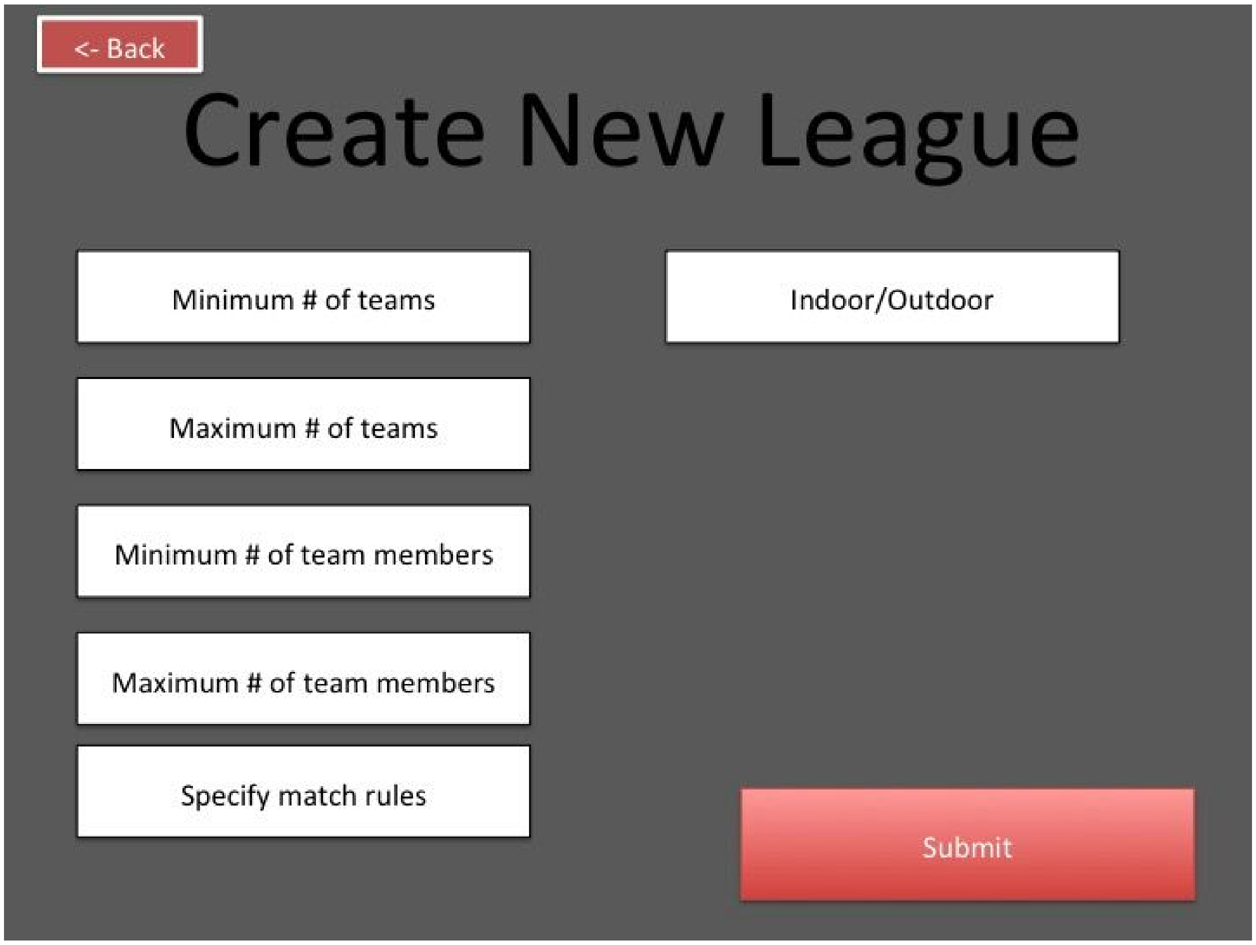
<- Back
Edit Existing Venue
Edit name
Edit address
Indoor/Outdoor
Delete Venue
Submit

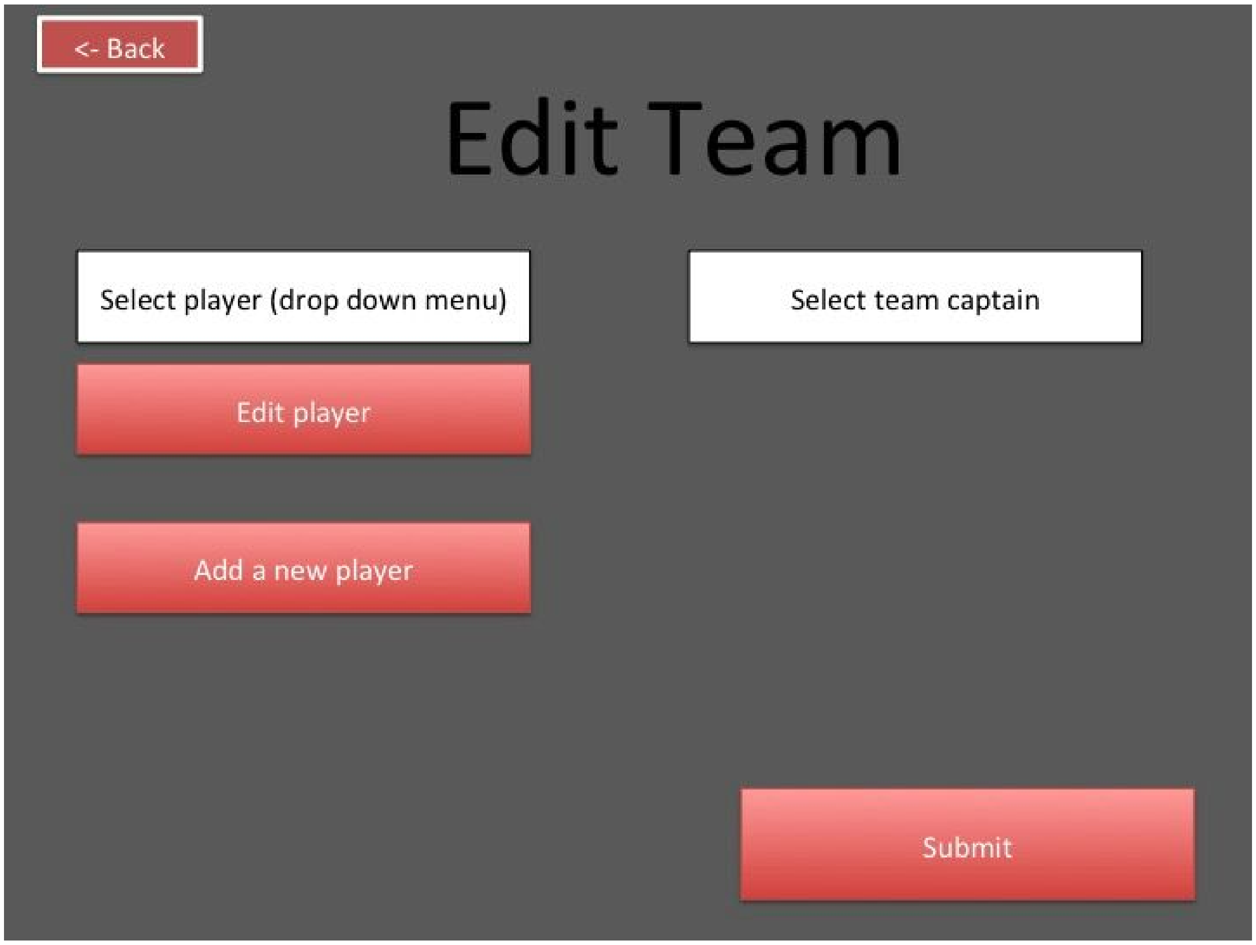

<- Back
Create New Venue
Name
Address
Indoor/Outdoor
Submit


<- Back
Leagues
Select an existing league (drop down menu)
Edit an existing league
Create a new league

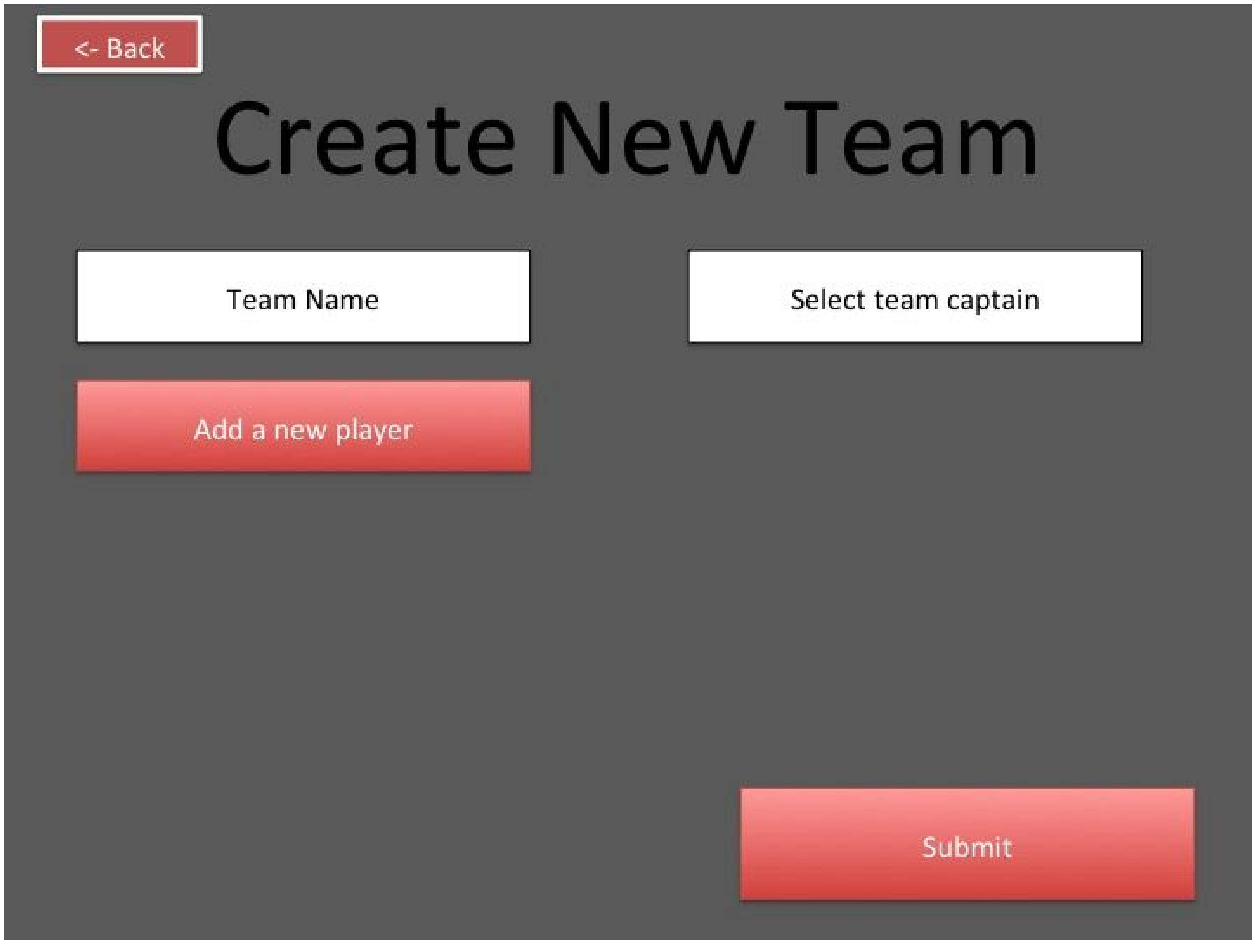


<- Back
Edit Existing League
Edit minimum # of teams
Edit maximum # of teams
Edit minimum # of team members
Edit maximum # of team members
Specify match rules
Cancel League
Edit Indoor/Outdoor
Select league team (drop down menu)
Edit team
Create new team
Submit

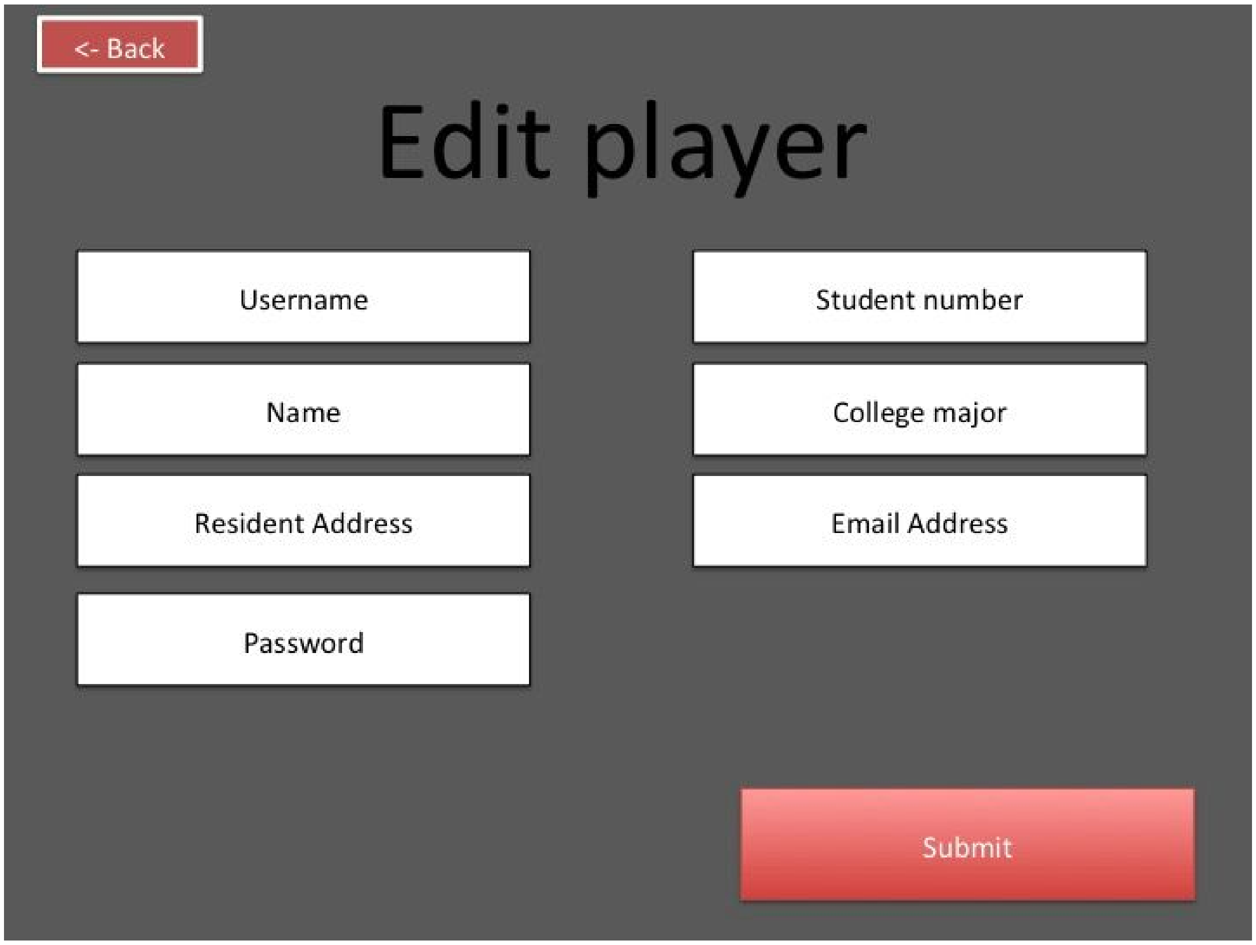

<- Back
Create New League
Minimum # of teams
Maximum # of teams
Minimum # of team members
Maximum # of team members
Specify match rules
Indoor/Outdoor
Submit



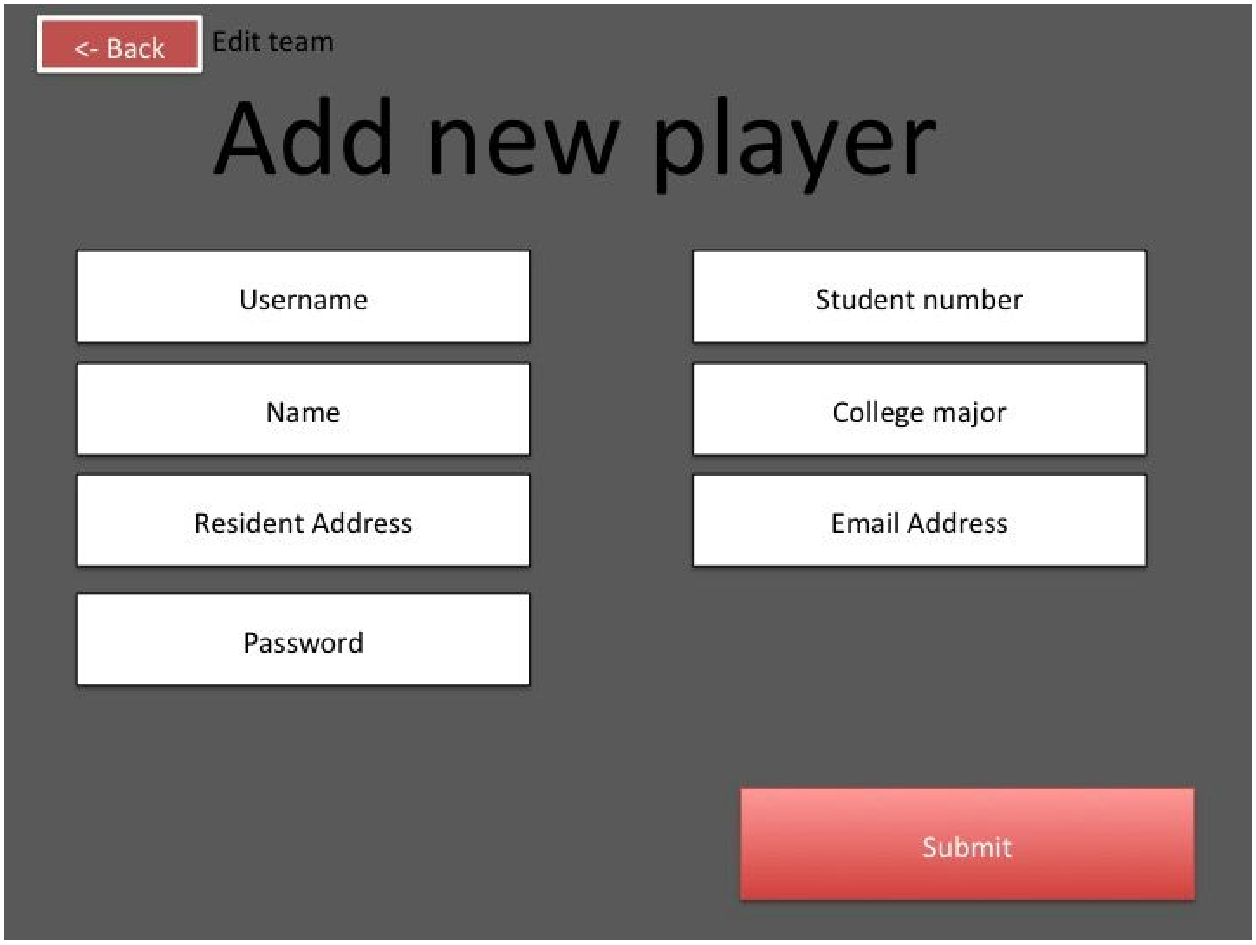
<- Back
Edit Team
Select player (drop down menu)
Select team captain
Edit player
Add a new player
Submit



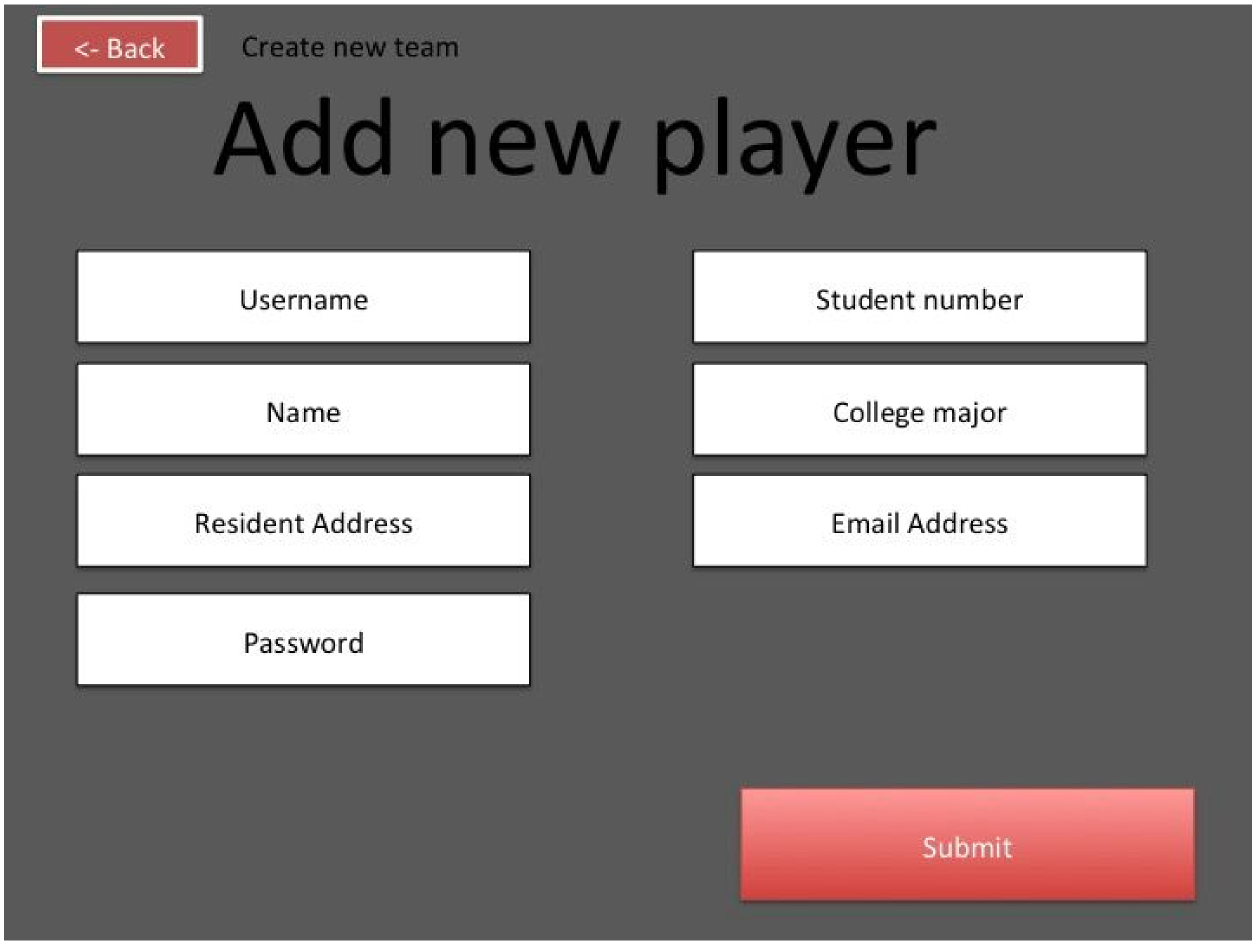
<- Back
Create New Team
Team Name
Select team captain
Add a new player
Submit



<- Back
Edit player
Username
Student number
Name
College major
Resident Address
Email Address
Password
Submit

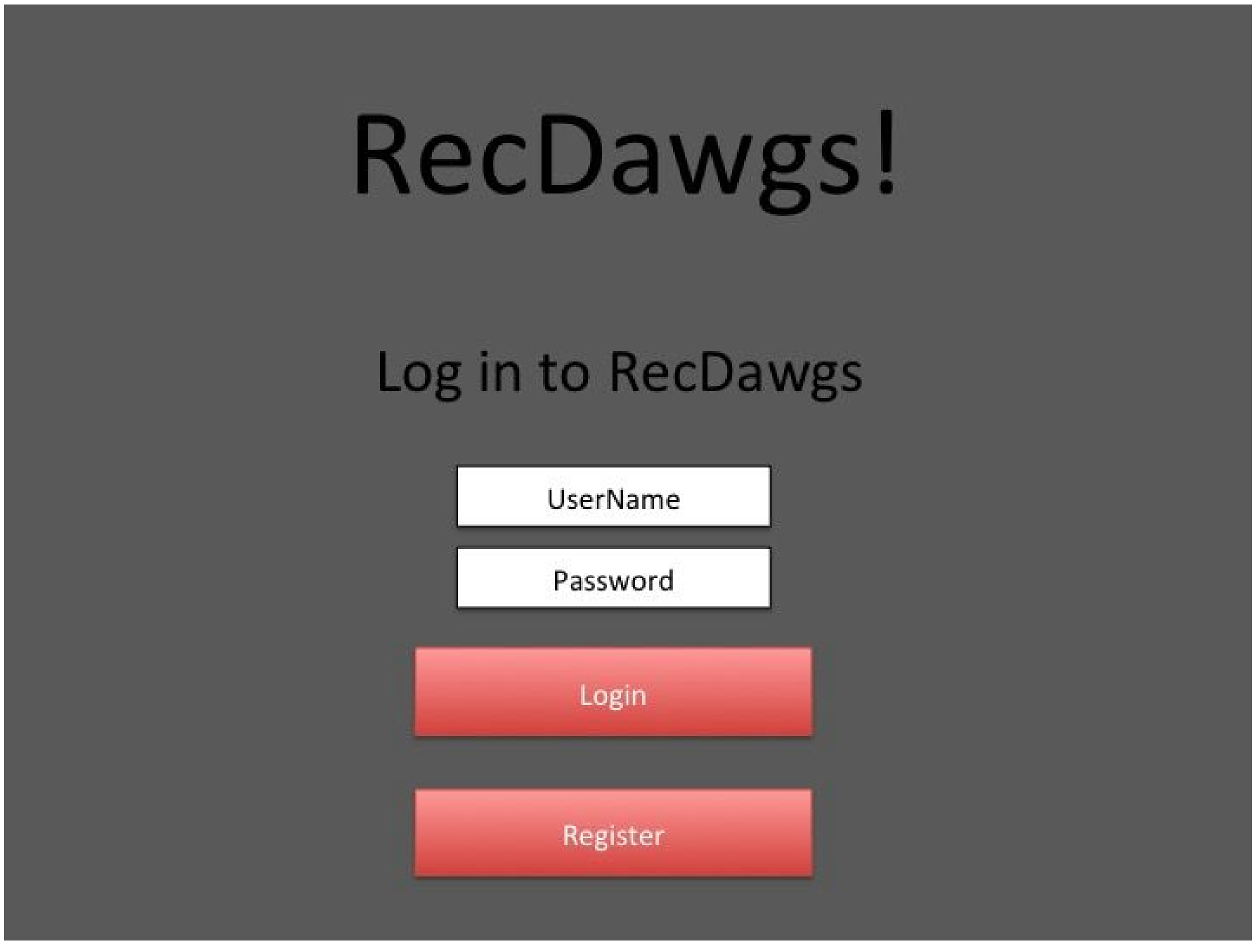


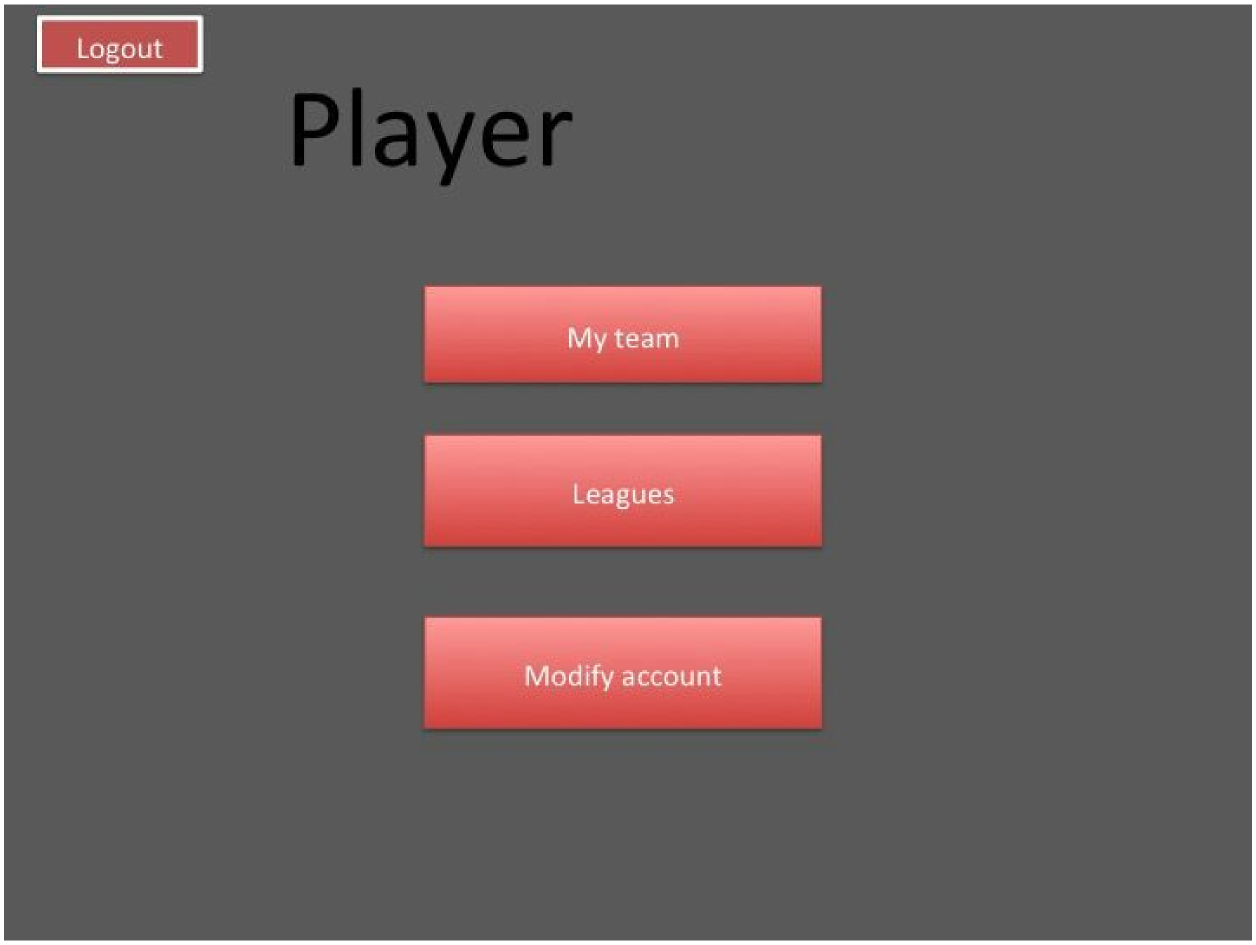
<- Back
Edit team
Add new player
Username
Name
Resident Address
Password
Student number
College major
Email Address
Submit

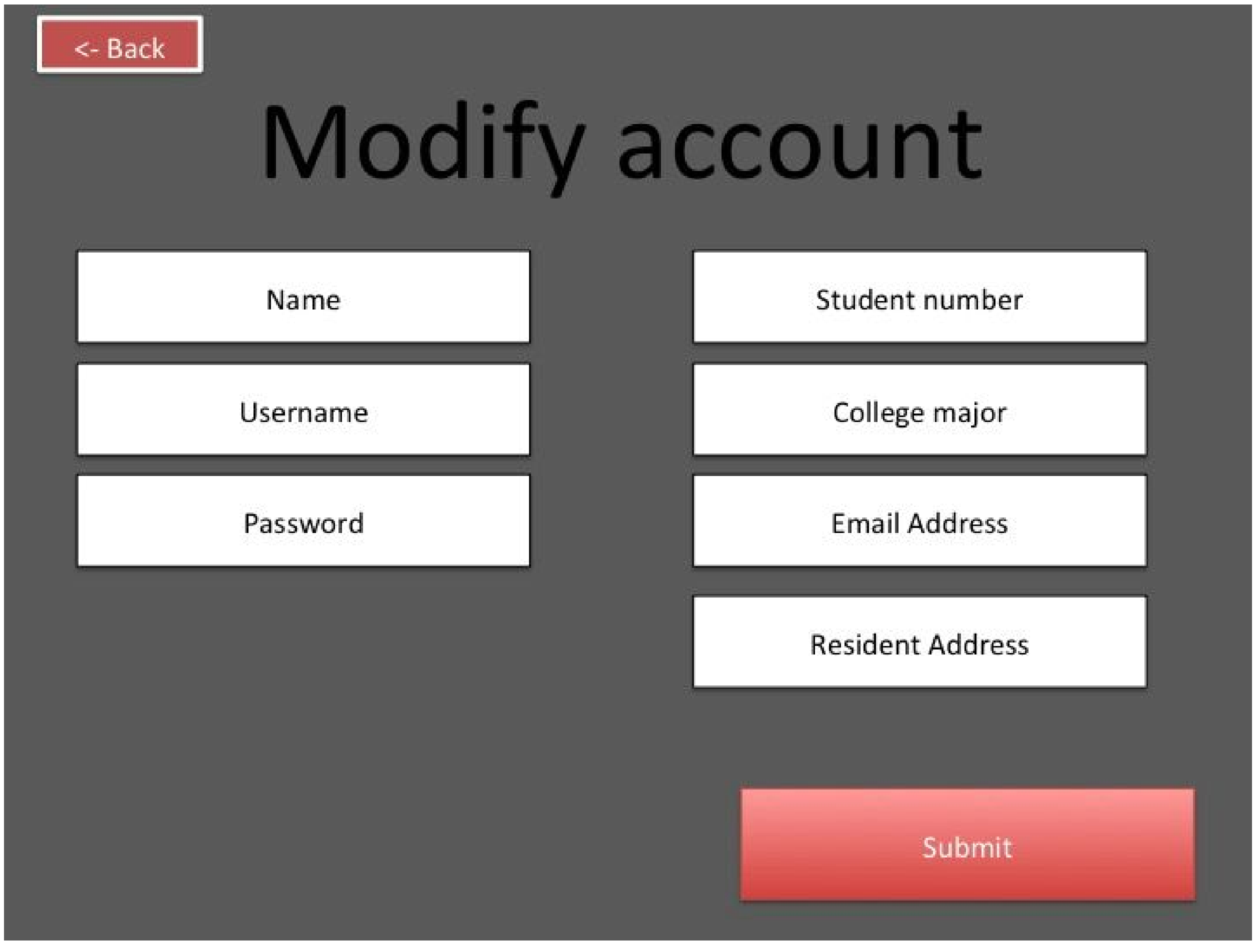


<- Back
Create new team
Add new player
Username
Student number
Name
College major
Resident Address
Email Address
Password
Submit

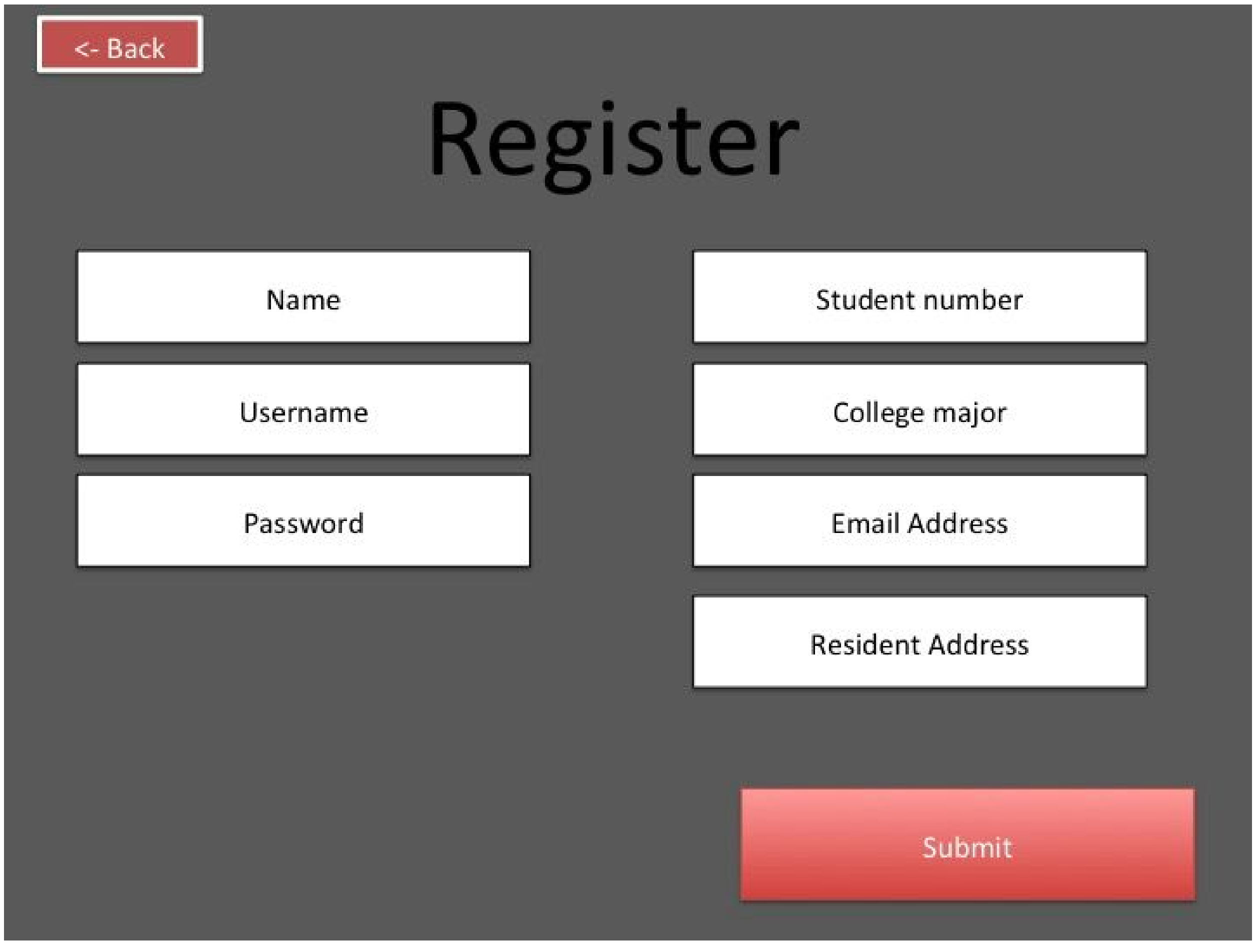
### 3.5.5.2 - User Interfaces: Player

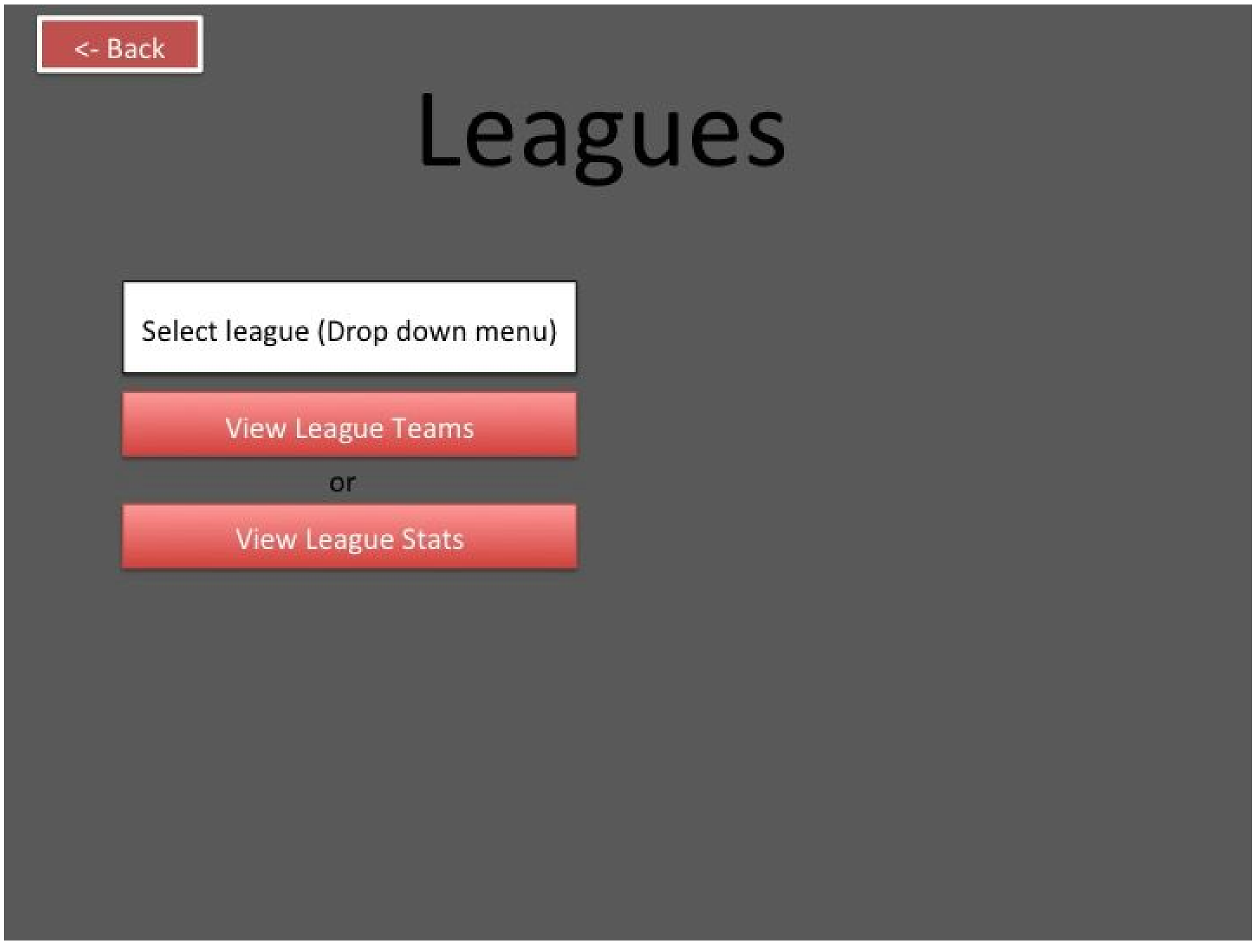



Logout
Player
My team
Leagues
Modify account

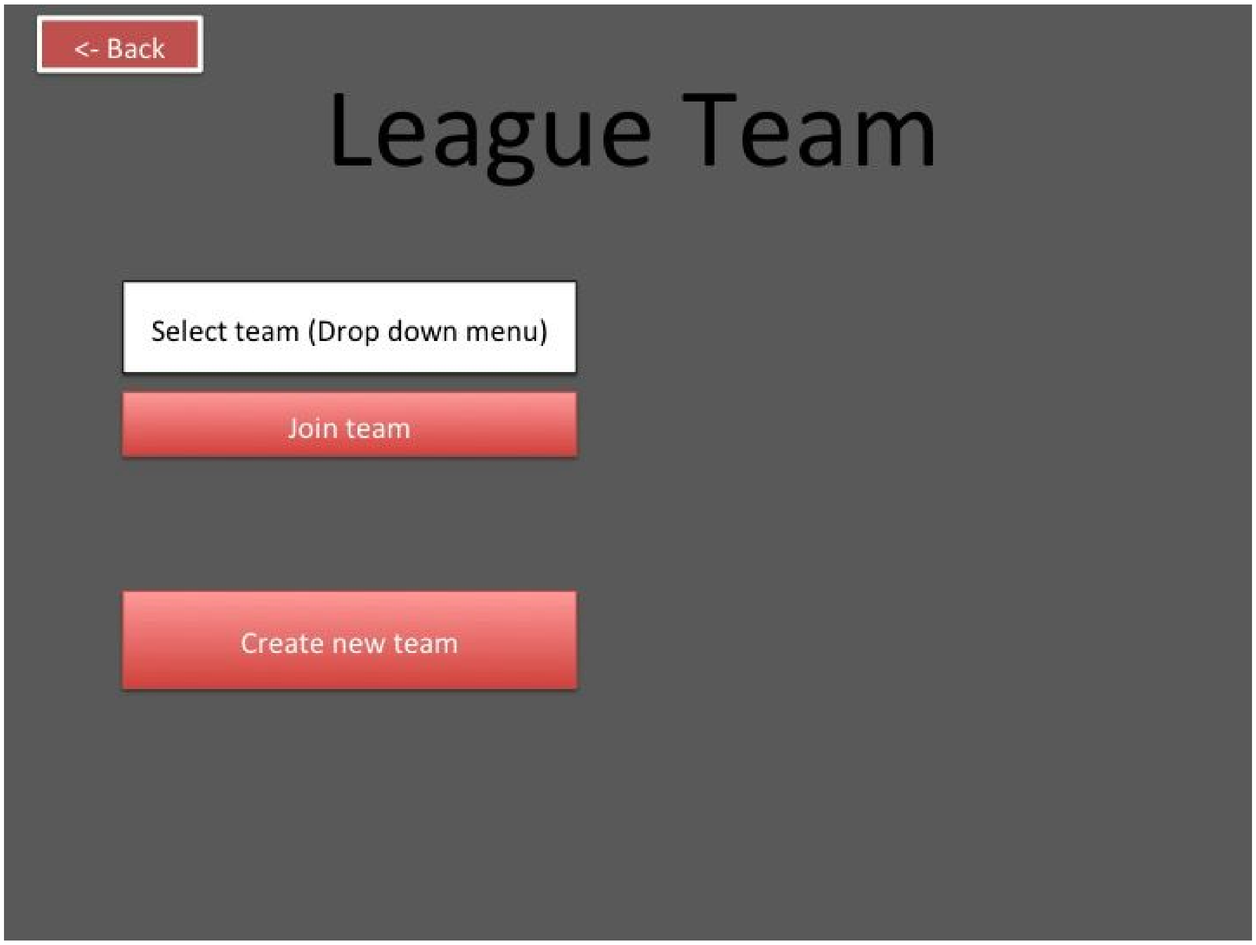


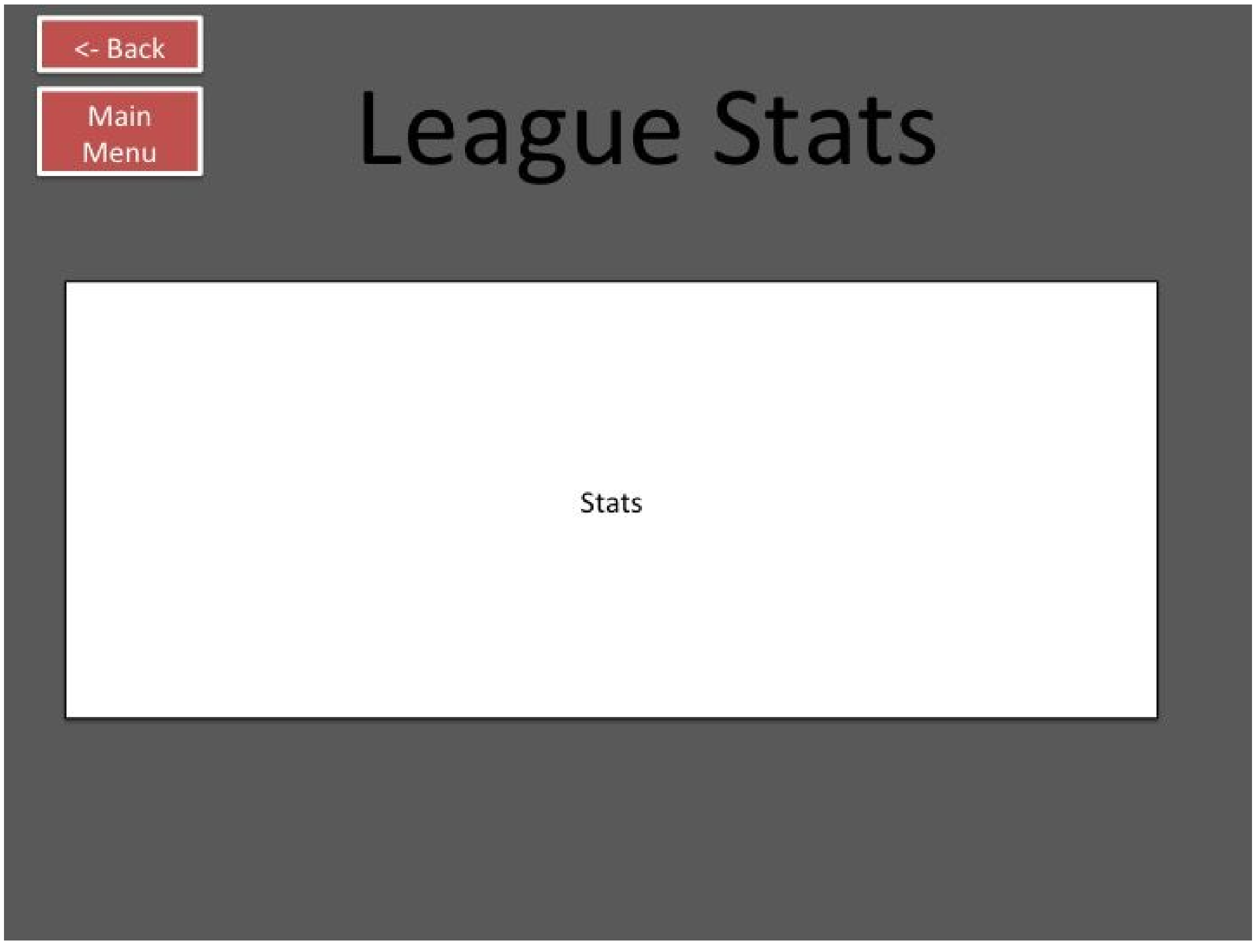
<- Back
Modify account
Name
Username
Password
Student number
College major
Email Address
Resident Address
Submit

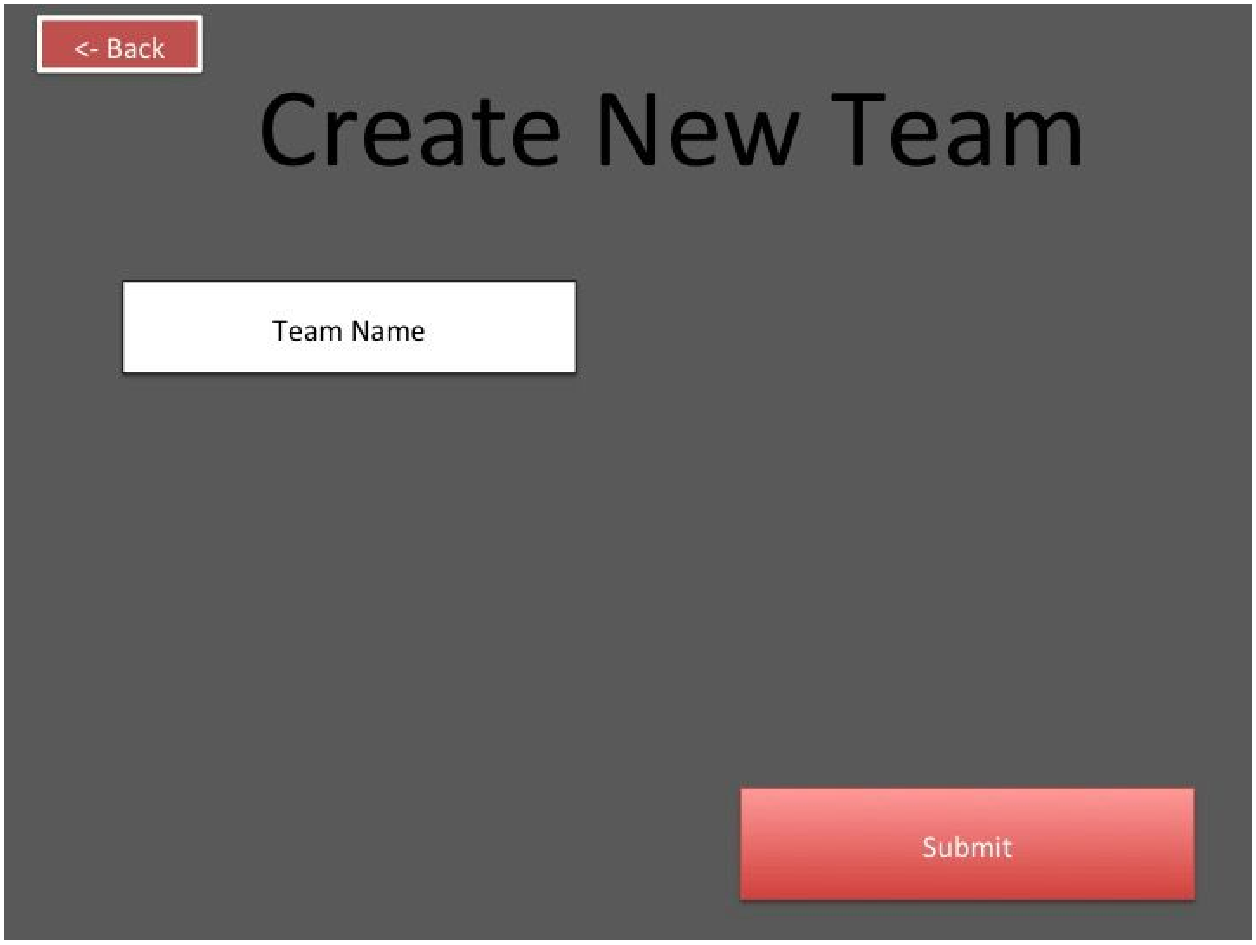

<- Back
Register
Name
Username
Password
Student number
College major
Email Address
Resident Address
Submit

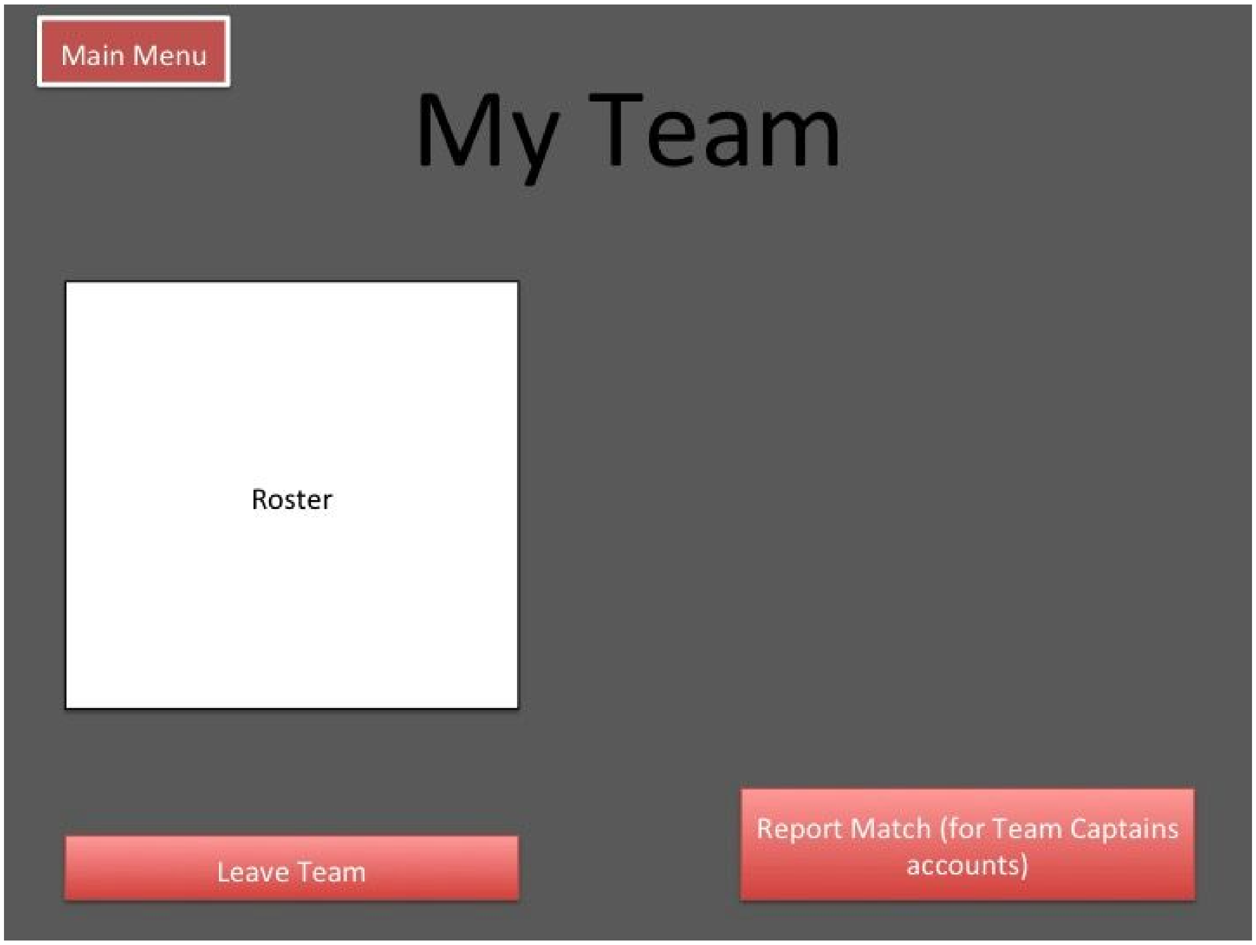


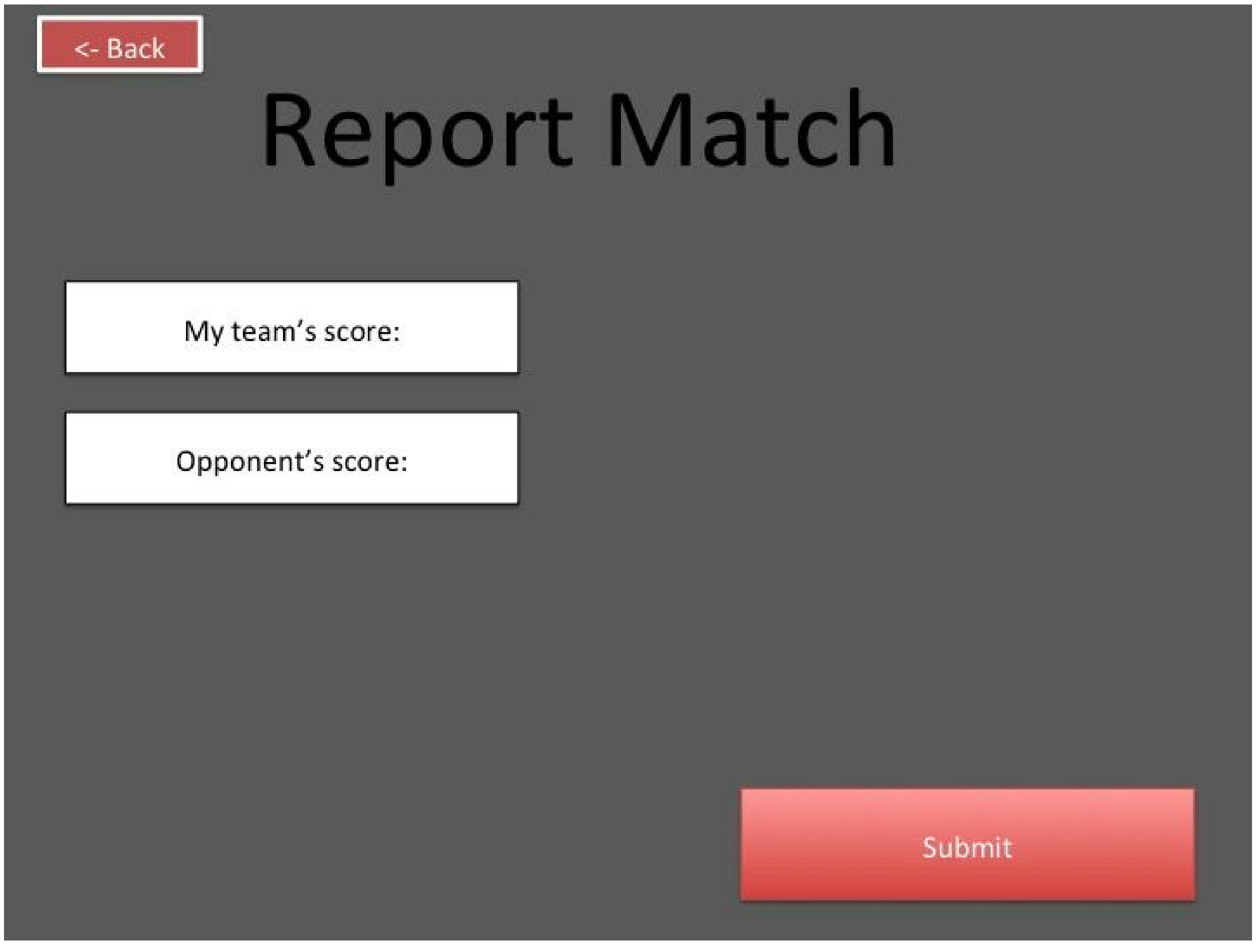
<- Back
Leagues
Select league (Drop down menu)
View League Teams
or
View League Stats


<- Back
League Team
Select team (Drop down menu)
Join team
Create new team


<- Back
Main Menu
League Stats
Stats


<- Back
Create New Team
Team Name
Submit


Main Menu
My Team
Roster
Leave Team
Report Match (for Team Captains accounts)



<- Back
Report Match
My team's score:
Opponent's score:
Submit

## Section 4: Glossary

| Word | Definition |
|---|---|
| Active league | A league in progress |
| Administrator | The person that manage RECDAWGS, Admins can create leagues/venues, appoint captains, etc. |
| League | A series of games, played in brackets, that allows the best team to win |
| Multi user access | A system that allows multiple users to access the system at the same time |
| Persistent data storage | Storing data and saving it so that it is not lost if the system goes down |
| Player | The default status of an account. This person have all the necessary options to play in a game. |
| RECDAWGS | A Recreational Sports Management Systems designed to support the management of several recreational sports leagues for a small college with minimal involvement of college staff. |
| Sports venues | Locations that the sport/sports will take place |
| System | This refers to the program, and what the program was created to accomplish/compute |
| Team Captain | The person that manage a team, Team captains have more power than players |
| User interface | The graphical interface presented to the user |