|  |
| --- |
| Sports Score Tracker |
| Design Specification Report |
| Cox, Christian - Jacob, Jason - Morrison, David |

Contents

[Introduction 5](#_Toc226770062)

[Problem Statement 5](#_Toc226770063)

[Proposal 5](#_Toc226770064)

[System Requirements 6](#_Toc226770065)

[Functional Requirements 6](#_Toc226770066)

[Non functional Requirements 10](#_Toc226770067)

[Use Cases 11](#_Toc226770068)

[Use Case Diagram 11](#_Toc226770069)

[Use Case Descriptions 12](#_Toc226770070)

[Use Case Validation 14](#_Toc226770071)

[Domain Explanation 16](#_Toc226770072)

[Class: User 16](#_Toc226770073)

[Class: Guest 16](#_Toc226770074)

[Class: Administrator 16](#_Toc226770075)

[Class: Sport 16](#_Toc226770076)

[Class: League 16](#_Toc226770077)

[Class: Team 16](#_Toc226770078)

[Class: Game 16](#_Toc226770079)

[Class: Predictions 16](#_Toc226770080)

[Class: Comment 16](#_Toc226770081)

[Domain Validation 17](#_Toc226770082)

[State Diagram 18](#_Toc226770083)

[Class Diagram 19](#_Toc226770084)

[Sequence Diagrams 20](#_Toc226770085)

[Sequence Diagram 1 – Register User 20](#_Toc226770086)

[Sequence Diagram 2 – Login User 21](#_Toc226770087)

[Sequence Diagram 3 – Logout User 22](#_Toc226770088)

[Sequence Diagram 4 – Modify Password 23](#_Toc226770089)

[Sequence Diagram 5 – Reset Password 24](#_Toc226770090)

[Sequence Diagram 6 – View Score 25](#_Toc226770091)

[Sequence Diagram 7 – View Schedule 26](#_Toc226770092)

[Sequence Diagram 8 – View Comments 27](#_Toc226770093)

[Sequence Diagram 9 – Post Comment 28](#_Toc226770094)

[Sequence Diagram 10 – View Predictions 29](#_Toc226770095)

[Sequence Diagram 11 – Publish Vote 30](#_Toc226770096)

[Sequence Diagram 12 – Create Sport 31](#_Toc226770097)

[Sequence Diagram 13 – Modify Sport 32](#_Toc226770098)

[Sequence Diagram 14 – Create League 33](#_Toc226770099)

[Sequence Diagram 15 – Modify League 34](#_Toc226770100)

[Sequence Diagram 16 – Create Team 35](#_Toc226770101)

[Sequence Diagram 17 – Modify Team 36](#_Toc226770102)

[Sequence Diagram 18 – Create Game 37](#_Toc226770103)

[Sequence Diagram 19 – Modify Game 38](#_Toc226770104)

[Sequence Diagram 20 – Modify User 40](#_Toc226770105)

[Sequence Diagram 21 – Delete Comment 41](#_Toc226770106)

[Activity Diagrams 42](#_Toc226770107)

[Database Diagrams 44](#_Toc226770108)

[Database Tables 44](#_Toc226770109)

[Database ER Model 45](#_Toc226770110)

[Conclusion 46](#_Toc226770111)

[Data Dictionary 47](#_Toc226770112)

[Class: Guest 47](#_Toc226770113)

[Class: User 47](#_Toc226770114)

[Class: Game 48](#_Toc226770115)

[Class: Predictions 50](#_Toc226770116)

[Class: Comment 50](#_Toc226770117)

[Class: Team 50](#_Toc226770118)

[Class: Administrator 51](#_Toc226770119)

[Class: League 52](#_Toc226770120)

[Class: Sport 53](#_Toc226770121)

# Introduction

## Problem Statement

The Southeastern Cal Ripken Baseball League based in Lexington, KY, needs to have a way to track their leagues via the World Wide Web. The current system in place calls for manual entry of all leagues and teams into a hand-created spreadsheet. Each year’s schedule is entered along with the outcome of each game. The records of each team must then be manually updated to reflect the scores. The standings of each league must then be manually updated to show which teams are currently ranked first, second, and so on. Additionally, the Southeastern Cal Ripken Baseball League also has several other age groups of leagues that work in conjunction with them. These leagues would also be interested in an online league tracking environment.

The Executive Board for the Southeastern Cal Ripken Baseball League would need to be able to add multiple leagues into an online tracking system of some sort. These leagues would need to be able to track multiple teams. Schedules, scores, and standings of these teams would also need to be able to be updated and viewed. The Executive Board would also like the coaches, parents, players, and even the general public to have the ability to view the schedules, scores, and standings for all of the teams in the Southeastern Cal Ripken Baseball league. Additionally, the Executive Board would need to be able to easily add other leagues and teams into the system.

## Proposal

The Sports Score Tracker is designed to function as a web-based application. The sports score tracker will be flexible enough to allow registered users to input their own data for their own custom sports, leagues, and teams. This data includes a full daily schedule, as well as league standings. Users will be able to click on individual games, and follow the score. Within the in-depth view of an individual game, users will also be able to post comments on this game. User comments will be filed under the tags of Injuries, Score, and Other. Users could choose to view only those comments within a given tag, or they can choose to view all comments. The site will not require users to login unless they would like to post comments, vote on outcomes of games, or manage their own teams/leagues. Additionally, within the chosen game, users can vote on which team they think will win. All users that access the site will be given the option to view the opinions of all users that voted via vote percentages corresponding to each game. Additionally, the sports score tracker will feature a smart system that calculates the most logical choice for the victor of the game. The smart outcome detection will take into account the winning percentage for each team. All users that access the site will be able to see the computer selection.

# System Requirements

## Functional Requirements

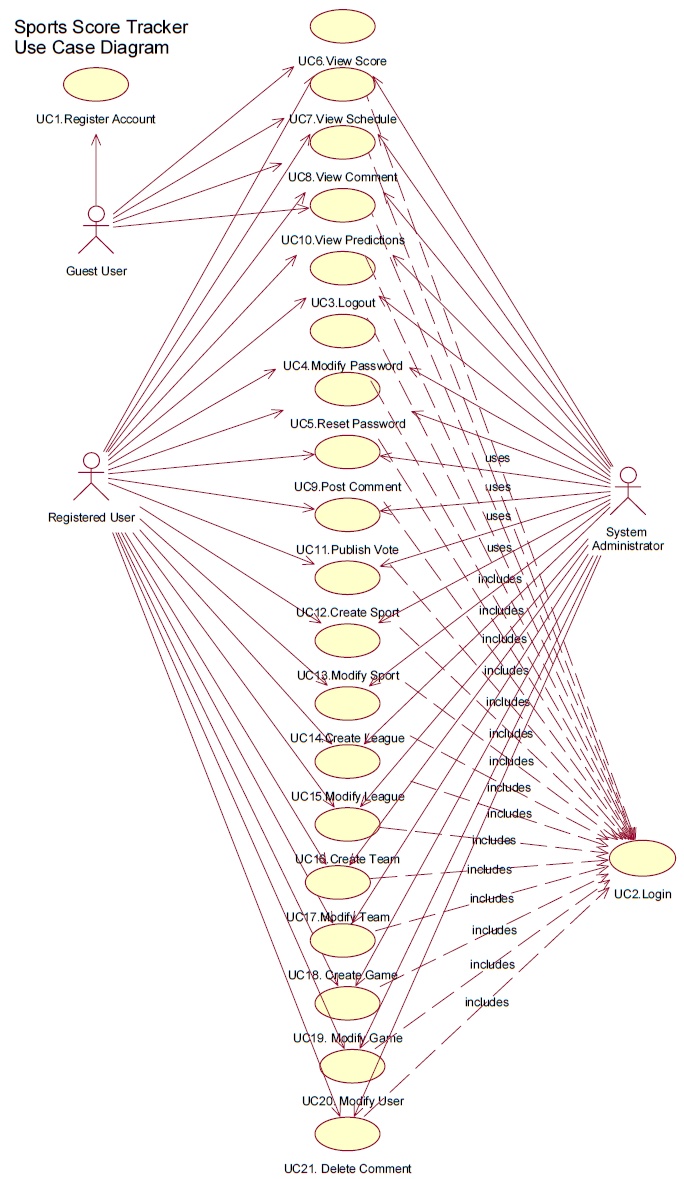
1. The system **must** provide a means for a guest to **register** for a new account**.**
   1. The guest **shall** be provided with the facilities to register for an account.
      1. The guest **must** provide a valid **email address**, **password**, and **first name** to register for an account.
      2. Upon registering, the system **must** ensure the email address is not already registered in the system.
      3. The email address entered **shall** be used as **account Id/login ID**.
      4. The user **shall** be provided with confirmation of their registration, via an email to the email address used for registration.
2. The system **must** provide a means for a registered user to **login/logout.**
   1. The user **shall** be provided with the facilities to login.
      1. The user **must** provide a valid **login ID** and **password** combination to login.
      2. The user **shall** be provided with facilities to confirm the request.
      3. The user **shall** be provided with confirmation that the request has been processed.
   2. The user **shall** be provided with the facilities to logout.
      1. The user **shall** be provided with facilities to logout after successful login.
3. The system **must** provide a means to **modify password.**
   1. Registered members **shall** be provided with facilities to **update current password**.
      1. The user **must** **login.**
      2. The user **must** supply current password and new password.
      3. The user **shall** be provided with facilities to confirm the request.
      4. The user **shall** be provided with confirmation that the request has been processed.
   2. The system **must** provide a means to **reset password.**
      1. Users **shall** be provided with facilities to **reset** their **password**.
      2. The user **shall** be provided with facilities to request a temporary password using their **email address**.
      3. Temporary password **shall** be emailed to the user email on file, if the login ID and email match.
      4. User **must login** with the temporary passwordand **update password**.
4. The system **must** provide a means to **view game scores.**
   1. Guests, registered users and administrators **shall** be able to view game scores.
   2. Game scores can be retrieved via league(s), sport(s) and/or teams.
5. The system **must** provide a means to **view game schedules.**
   1. Guests, registered users and administrators **shall** be able to view game schedules.
   2. Schedules can be retrieved via league(s), sport(s) and/or teams.
6. The system **must** provide a means to **view comments.**
   1. Guests, registered users and administrators **shall** be able to view comments per game.
7. The system **must** provide a means to **post** **comments.**
   1. Registered users and administrators **shall** be able to post comments per game.
   2. The **user** must **login** to be able to post comments.
8. The system **must** provide a means to vote which team the user predicts will win the game**.**
   1. Registered users and administrators **shall** be provided with facilities to vote per game.
   2. The **user** must **login** to be able to vote.
9. The system must provide a means to **view** the system and user **estimated** **winning** **team** predictions.
   1. Guests, registered users and administrators **shall** be able to view the system prediction and the user predicted winning team.
   2. The system prediction **shall** take into account the winning percentage for each team.
   3. The system **shall** sum the user votes to view the user estimated winning team.
      1. The results **shall** be represented as a percentage chart that will show which team in a game will win.
10. The system **must** provide a means to **create / modify** ***custom*** **sports.**
    1. Registered users and administrators **shall** be provided with the facilities to **create** *custom* sports.
       1. Theuser **must** login to be able to create *custom* sports.
       2. Theuser **shall** be provided with the facilities to input a **sport name**.
       3. The user **shall** be provided with facilities to confirm the request.
       4. The user **shall** be provided with confirmation that the request has been processed.
    2. Administrators **shall** be provided with the facilities to **delete** *custom* sports.
       1. Theuser **must** login to be able to **delete** *custom* sports.
       2. Theuser **shall** be provided with the facilities to select a sport for deletion by sport name.
       3. The user shall be informed that all leagues, teams, games, and comments related information to the sport will also be deleted.
       4. The user **shall** be provided with facilities to confirm the request to delete the sport.
       5. The user **shall** be provided with confirmation that the request has been processed (successfully deleted).
11. The system **must** provide a means to **create/modify** ***custom*** **leagues.**
    1. Registered users and administrators **shall** be provided with the facilities to **create** *custom* leagues.
       1. Theuser **must** login to be able to create *custom* leagues.
       2. Theuser **shall** be provided with the facilities to select a sport and input a new **league name**.
       3. The user **shall** be provided with facilities to confirm the request.
       4. The user **shall** be provided with confirmation that the request has been processed.
    2. Registered users and administrators **shall** be provided with the facilities to **update** *custom* leagues.
       1. Theuser **must** login to be able to update *custom* leagues.
       2. Theuser **shall** be provided with the facilities to look up a custom league by league name.
       3. Theuser **shall** be provided with the facilities to input a new league name.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
    3. Registered users and administrators **shall** be provided with the facilities to **delete** *custom* leagues.
       1. Theuser **must** login to be able to **delete** *custom* leagues.
       2. Theuser **shall** be provided with the facilities to select a league for deletion by league name.
       3. The user shall be informed that all teams, games, and comments related information to the league will also be deleted.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
12. The system **must** provide a means to **create/modify** ***custom*** **teams.**
    1. Registered users and administrators **shall** be provided with the facilities to **create** *custom* teams.
       1. Theuser **must** login to be able to create *custom* teams.
       2. Theuser **shall** be provided with the facilities to select a sport and input a new **team name**, and join the team into an existing **league**.
       3. The user **shall** be provided with facilities to confirm the request for adding a new team.
       4. The user **shall** be provided with confirmation that the request has been processed.
    2. Registered users and administrators **shall** be provided with the facilities to **update** *custom* teams.
       1. Theuser **must** login to be able to update *custom* teams.
       2. Theuser **shall** be provided with the facilities to look up a custom team by team name.
       3. Theuser **shall** be provided with the facilities to input a new team name and modify the league.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
    3. Registered users and administrators **shall** be provided with the facilities to **delete** *custom* teams.
       1. Theuser **must** login to be able to **delete** *custom* teams.
       2. Theuser **shall** be provided with the facilities to select a team for deletion by team name.
       3. The user shall be informed that all games and comments related information to the team will also be deleted.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
13. The system **must** provide a means to **create/modify** ***custom*** **games.**
    1. Registered users and administrators **shall** be provided with the facilities to **create** *custom* games.
       1. Theuser **must** login to be able to create *custom* games.
       2. Theuser **shall** be provided with the facilities to select two teams and input the game details of date, home score and away score if known.
       3. The user **shall** be provided with facilities to confirm the request for adding a new game.
       4. The user **shall** be provided with confirmation that the request has been processed.
    2. Registered users and administrators **shall** be provided with the facilities to **update** *custom* games.
       1. Theuser **must** login to be able to update *custom* games.
       2. Theuser **shall** be provided with the facilities to view games.
       3. Theuser **shall** be provided with the facilities to update the home and away score, and date.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
    3. Registered users and administrators **shall** be provided with the facilities to **delete** *custom* games.
       1. Theuser **must** login to be able to **delete** *custom* games.
       2. Theuser **shall** be provided with the facilities to select a game for deletion.
       3. The user shall be informed that all comments related to the game will also be deleted.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.
14. The system **must** provide a means to **delete** **comments.**
    1. Administrators **shall** be able to delete comments per game.
    2. The **user** must **login** to be able to delete comments.
15. The system must provide a means to modify a user.
    1. Administrators shall be provided with the facilities to modify a user.
    2. Theuser **must** login to be able to modify users.
       1. Theuser **shall** be provided with the facilities to look up a user by email address.
       2. Theuser **shall** be provided with the facilities to input a new email address, login ID, and/or password.
       3. The user **shall** be provided with facilities to confirm the request.
       4. The user **shall** be provided with confirmation that the request has been processed.
    3. Administrators **shall** be provided with the facilities to **delete** users.
       1. Theuser **must** login to be able to **delete** *users*.
       2. The user **shall** be provided with the facilities to select a user for deletion.
       3. The user shall be informed that all leagues, teams, games, and comments related to the user will also be deleted.
       4. The user **shall** be provided with facilities to confirm the request.
       5. The user **shall** be provided with confirmation that the request has been processed.

## Non functional Requirements

1. The system **shall** **ensure that login credentials meet minimum security specifications**
   1. The system **shall ensure that emails meet minimum requirements.**
      1. The system **shall** **ensure** email addresses are unique.
      2. The system **shall** **ensure** email addresses adhere to the standard Fully Qualified Domain Address format - *username*@*domain***.**
   2. The system **shall ensure that passwords meet minimum requirements.**
      1. The system **shall** **ensure** passwords are at least 8 characters in length.
      2. The system **shall** **ensure** passwords contain alphanumeric and special characters.
2. The system **shall** **ensure that comments are filtered before posting.**
   * 1. The system **shall** **ensure** comments do not exceed the 300 character limit.
3. The system **shall** **ensure that custom teams can only be modified by creator.**
4. The system **shall** **ensure that custom sports can only be modified by creator.**
5. The system **shall** **ensure that custom leagues can only be modified by creator.**
6. The system **shall ensure that custom games can only be modified by creator.**

# Use Cases

## Use Case Diagram

****

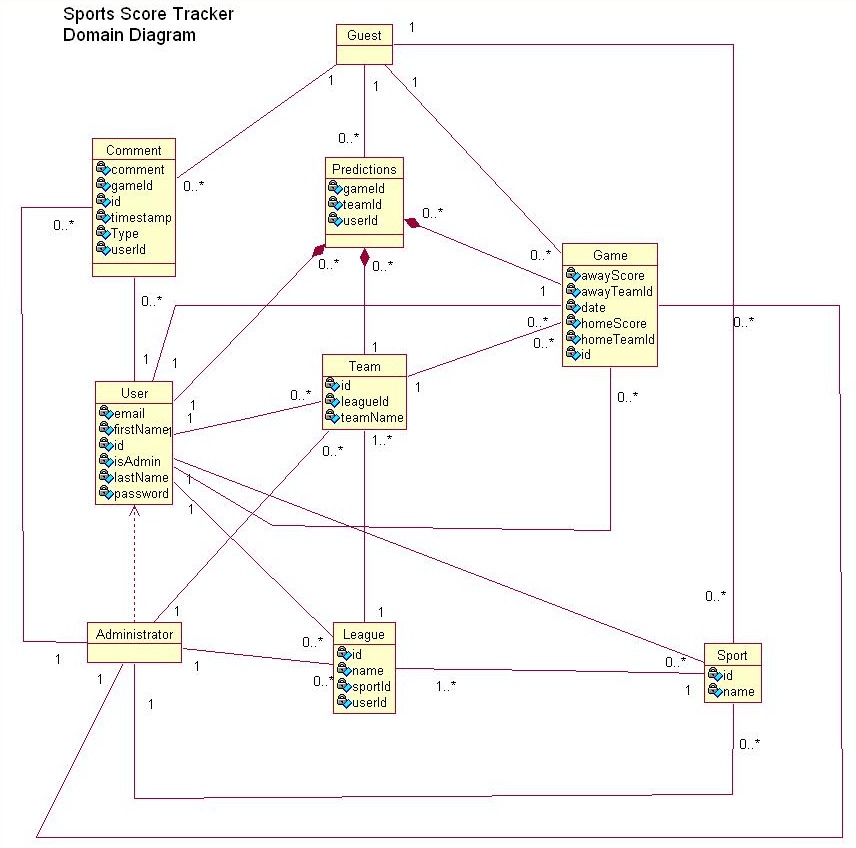
## Use Case Descriptions

|  |  |  |
| --- | --- | --- |
| **UC1.** | *Register User* | This system use is intended for guests to become registered members receive member privileges such as voting, creating and maintaining custom sports, leagues and teams.  **Dependencies: None.** |
| **UC2.** | *Login User* | This system use is intended for registered users to gain access to those modules of the system which require registration to use.  **Dependencies: UC1.** |
| **UC3.** | *Logout User* | This system use is intended for registered users to exit the system. User must first be logged in to access this option.  **Dependencies: UC2.** |
| **UC4.** | *Modify Password* | This system use is intended for registered users to update their current password and requires.  **Dependencies: UC2.** |
| **UC5.** | *Reset Password* | This system use is intended for users to request that the password for an account be reset and emailed to the account using the account id.  **Dependencies: None** |
| **UC6.** | *View Score* | This system use is intended for users to view game scores.  **Dependencies: None** |
| **UC7.** | *View Schedule* | This system use is intended for users to view game schedules.  **Dependencies: None** |
| **UC8.** | *View Comment* | This system use is intended for users to view user comments.  **Dependencies: None** |
| **UC9.** | *Post Comment* | This system use is intended for registered users and administrators to post comments.  **Dependencies: UC2.** |
| **UC10.** | *View Predictions* | This system use is intended for guests, registered users, and administrators to view the predicted estimated winners based on user prediction and system prediction.  **Dependencies: None** |
| **UC11.** | *Publish Vote* | This system use is intended for registered users and administrators to post comments.  **Dependencies: UC2.** |
| **UC12.** | *Create Sport* | This system use is intended for registered users and administrators to create a custom sport.  **Dependencies: UC2.** |
| **UC13.** | *Modify Sport* | This system use is intended for registered users and administrators to update or delete a custom sport created by that user.  **Dependencies: UC11.** |
| **UC14.** | *Create League* | This system use is intended for registered users and administrators to create a custom league.  **Dependencies: UC2.** |
| **UC15.** | *Modify League* | This system use is intended for registered users and administrators to update or delete a custom league created by that user.  **Dependencies: UC13.** |
| **UC16.** | *Create Team* | This system use is intended for registered users and administrators to create a custom team.  **Dependencies: UC2.** |
| **UC17.** | *Modify Team* | This system use is intended for registered users and administrators to update or delete a custom team created by that user.  **Dependencies: UC15.** |
| **UC18.** | *Create Game* | This system use is intended for registered users and administrators to create a custom game.  **Dependencies: UC2.** |
| **UC19.** | *Modify Game* | This system use is intended for registered users and administrators to update or delete a custom game for teams in a league owned by that user.  **Dependencies: UC15.** |
| **UC20.** | *Modify User* | This system use is intended for administrators to update or delete a user.  **Dependencies: UC2.** |
| **UC21.** | *Delete Comment* | This system use is intended for administrators to delete a comment.  **Dependencies: UC2.** |
|  |  |  |

## Use Case Validation

**Requirements vs. Use Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **R1** | **R2** | **R3** | **R4** | **R5** | **R6** | **R7** | **R8** | **R9** | **R10** | **R11** | **R12** | **R13** | **R14** | **R15** |
|  |  | Register New Account | Login/Logout | Modify Password | View Game Scores | View Game Schedules | View Comments | Post Comments | Publish Vote | View Predictions | Create / Modify Sport | Create / Modify League | Create / Modify Team | Create / Modify Game | Delete Comment | Modify User |
| **UC1** | Register Account | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **UC2** | Login user | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **UC3** | Logout user |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **UC4** | Modify Password |  | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| **UC5** | Reset password |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| **UC6** | View Score |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |
| **UC7** | View Schedule |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |
| **UC8** | View Comment |  |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |
| **UC9** | Post Comment |  |  |  |  |  |  | **X** |  |  |  |  |  |  |  |  |
| **UC10** | View Predictions |  |  |  |  |  |  |  |  | **X** |  |  |  |  |  |  |
| **UC11** | Publish Vote |  | **X** |  |  |  |  |  | **X** |  |  |  |  |  |  |  |
| **UC12** | Create Sport |  | **X** |  |  |  |  |  |  |  | **X** |  |  |  |  |  |
| **UC13** | Modify Sport |  | **X** |  |  |  |  |  |  |  | **X** |  |  |  |  |  |
| **UC14** | Create League |  | **X** |  |  |  |  |  |  |  |  | **X** |  |  |  |  |
| **UC15** | Modify League |  | **X** |  |  |  |  |  |  |  |  | **X** |  |  |  |  |
| **UC16** | Create Team |  | **X** |  |  |  |  |  |  |  |  |  | **X** |  |  |  |
| **UC17** | Modify Team |  | **X** |  |  |  |  |  |  |  |  |  | **X** |  |  |  |
| **UC18** | Create Game |  | **X** |  |  |  |  |  |  |  |  |  |  | **X** |  |  |
| **UC19** | Modify Game |  | **X** |  |  |  |  |  |  |  |  |  |  | **X** |  |  |
| **UC20** | Modify User |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  | **X** |
| **UC21** | Delete Comment |  | **X** |  |  |  |  |  |  |  |  |  |  |  | **X** |  |



## Domain Explanation

**Class: User**

The User object stores all attributes, functions, and operations related to the user.

**Class: Guest**

The Member object is used to store attributes, functions, and operations related to a member. This object will inherit properties of the User, but with more attributes and operations that require elevated privileges.

**Class: Administrator**

The Administrator object is used to store attributes, functions, and operations related to the administrator. This object will inherit properties of the User, but with more operations that require elevated privileges.

**Class: Sport**

The Sport is used to store attributes, functions, and operations related to a Sport.

**Class: League**

The League is used to store attributes, functions, and operations related to a League.

**Class: Team**

The Team is used to store attributes, functions, and operations related to a Team.

**Class: Game**

The Game is used to store attributes, functions, and operations related to a Game.

**Class: Predictions**

The Prediction is used to store attributes, functions, and operations related to a Prediction.

**Class: Comment**

The Comment is used to store attributes, functions, and operations related to a Comment.

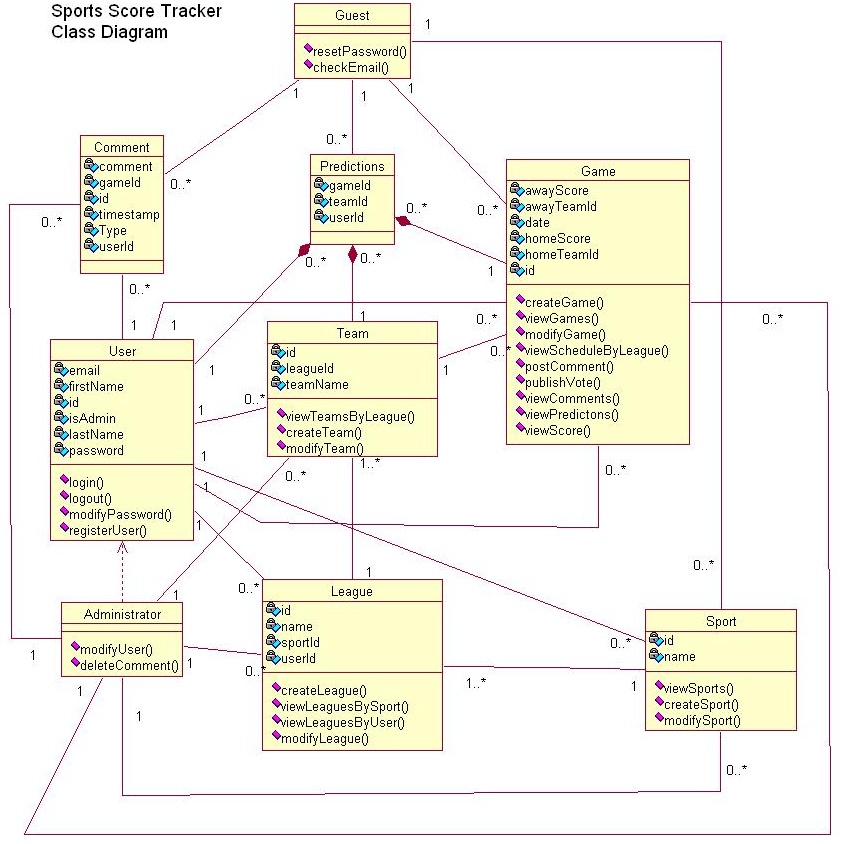
### Domain Validation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** | **R4** | **R5** | **R6** | **R7** | **R8** | **R9** | **R10** | **R11** | **R12** | **R13** | **R14** | **R15** |
|  | Register New Account | Login/Logout | Modify Password | View Game Scores | View Game Schedules | View Comments | Post Comments | Publish Vote | View Predictions | Create / Modify Sport | Create / Modify League | Create / Modify Team | Create / Modify Game | Delete Comment | Modify User |
| **Class: User** |  | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **Class: Guest** | **X** |  | **X** | **X** | **X** | **X** |  |  | **X** |  |  |  |  |  |  |
| **Class: Administrator** |  | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** |
| **Class: Sport** |  |  |  | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** | **X** |
| **Class: League** |  |  |  | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** | **X** | **X** | **X** | **X** |
| **Class: Team** |  |  |  | **X** | **X** | **X** | **X** | **X** | **X** |  |  | **X** | **X** | **X** | **X** |
| **Class: Game** |  |  |  | **X** | **X** | **X** | **X** | **X** | **X** |  |  |  | **X** | **X** | **X** |
| **Class: Prediction** |  |  |  |  |  |  |  | **X** | **X** |  |  |  | **X** |  | **X** |
| **Class: Comment** |  |  |  |  |  | **X** | **X** |  |  |  |  |  | **X** | **X** | **X** |

# State Diagram

## C:\Users\David\Desktop\My Projects\CSC835 - Team Project\Documents\Activity State Diagrams\stateDiagram.JPG

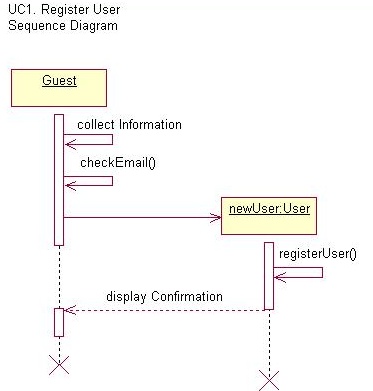
# Class Diagram

****

# Sequence Diagrams

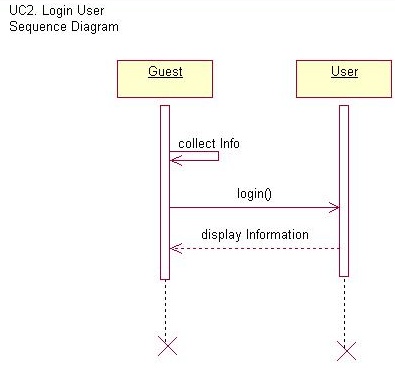
### Sequence Diagram 1 – Register User

The scenario of when a guest chooses to register as a user. Information such as email and password is collected from the guest. The email is verified against existing records to make sure that it is unique. A new User record is created and registered with the system. Confirmation of registration result is displayed back to the user.



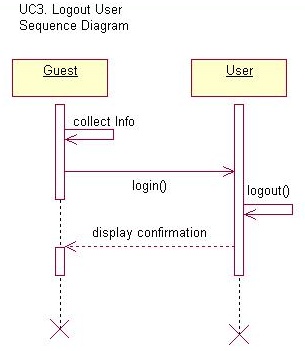
### Sequence Diagram 2 – Login User

The scenario of when a user logs in to gain access to the system. Information such as email and password is collected and the system authenticates these credentials against existing records. Confirmation of login results is displayed back to the user.



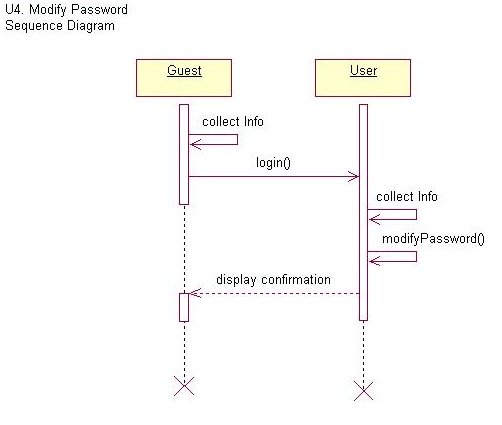
### Sequence Diagram 3 – Logout User

The scenario of when a user logs out of the system.A user, who has successfully logged in, has access to logout. Confirmation of a logout results is displayed back to the user.



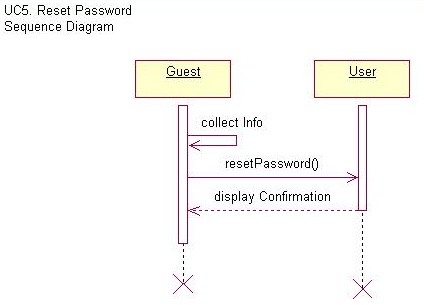
### Sequence Diagram 4 – Modify Password

The scenario of when a user modifies their password.A user, who has successfully logged in, has access to modify their password. Old and new password details are collected from the user and the current password for the user is updated. Confirmation of a password modification results is displayed back to the user. *(For more information on login see Sequence Diagram 2)*



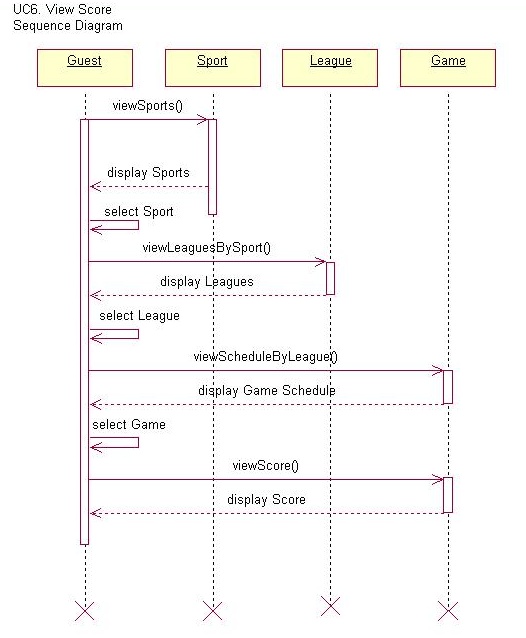
### Sequence Diagram 5 – Reset Password

The scenario of when a user resets their password.A user can reset their password without logging in to the system. The user email is collected and a randomly generated password is sent to it. Confirmation of the password reset results is displayed back to the user.

****

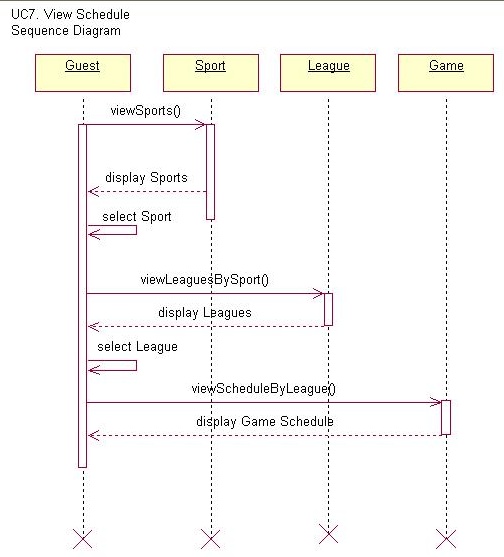
### Sequence Diagram 6 – View Score

The scenario of when a guest/user views a game score. A request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A game is selected from the list of games scheduled for the league. A request is made to view the Score for the selected game. A display of the score for the selected game is returned.



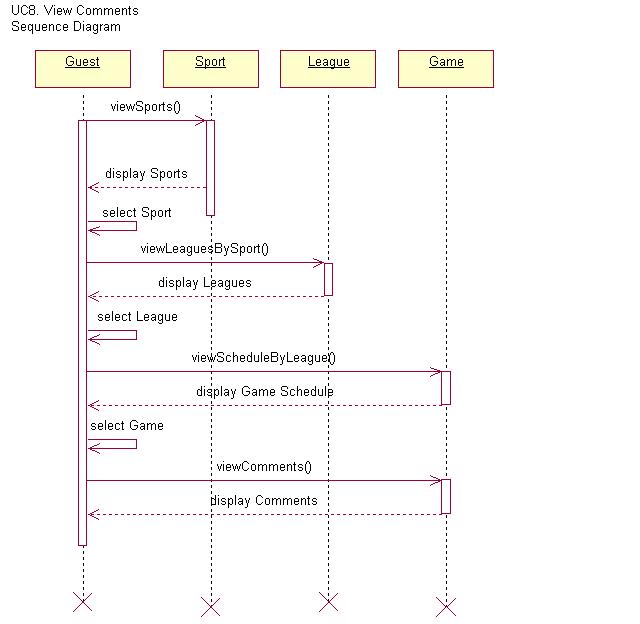
### Sequence Diagram 7 – View Schedule

The scenario of when a guest/user views a game schedule. A request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A display of the game schedule for the selected league is returned.



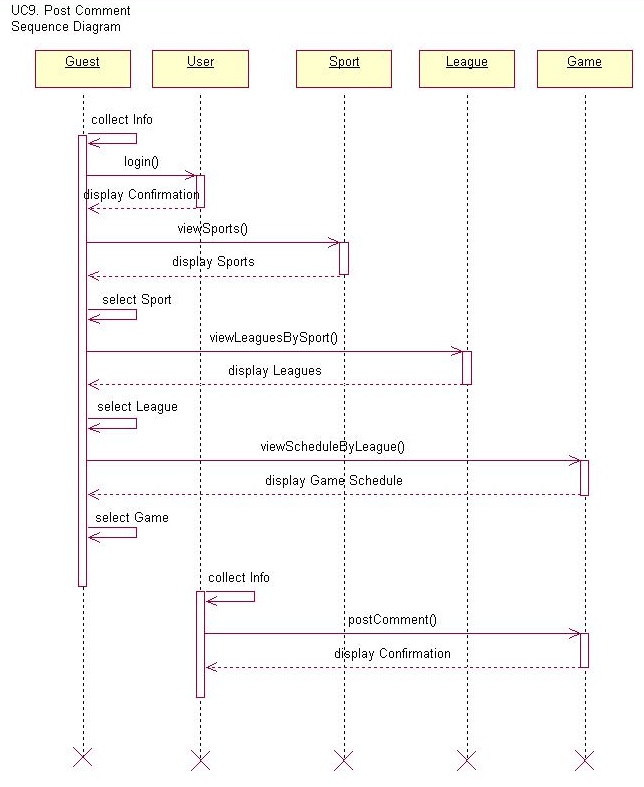
### Sequence Diagram 8 – View Comments

The scenario of when a guest/user views a game’s comments. A request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A game is selected from the list of games scheduled for the league. A request is made to view the comments for the selected game. A display of the comments for the selected game is returned.



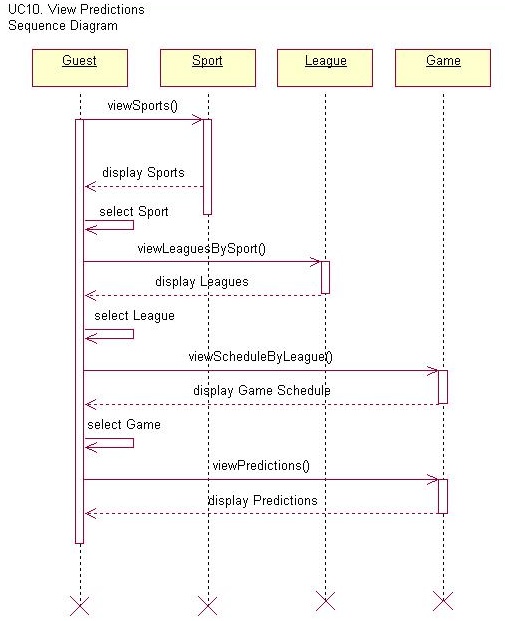
### Sequence Diagram 9 – Post Comment

The scenario of when a user posts a game comment. A user must first be logged in to post comments. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A game is selected from the list of games scheduled for the league. Comment details are collected from the user and a request is made to post the comment for the selected game. A display of the confirmation of the posted comment is returned.



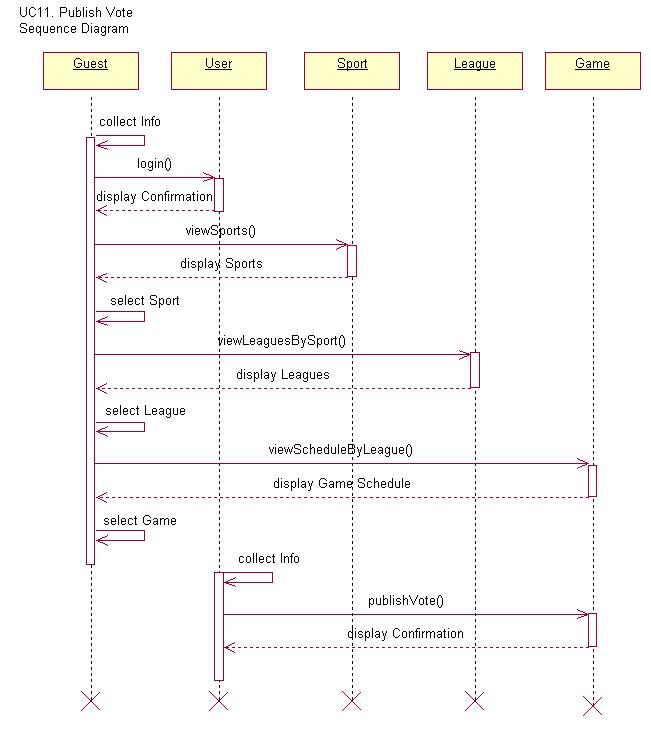
### Sequence Diagram 10 – View Predictions

The scenario of when a guest/user views system estimated game predictions. A request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A game is selected from the list of games scheduled for the league. A request is made to view the system estimated predictions for the selected game. A display of the predictions for the selected game is returned.



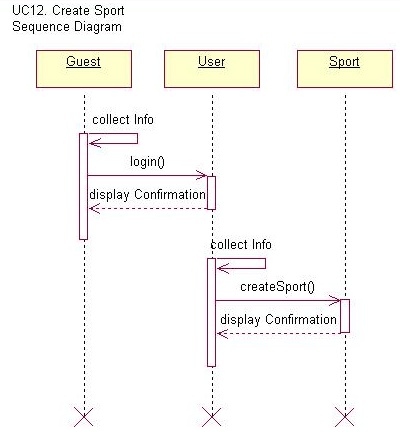
### Sequence Diagram 11 – Publish Vote

The scenario of when a user publishes a vote for a game. A user must first be logged in to publish a vote. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view the game schedule for the selected league. A game is selected from the list of games scheduled for the league. Vote details are collected from the user and a request is made to publish the vote for the selected game. A display of the confirmation of the published vote is returned.



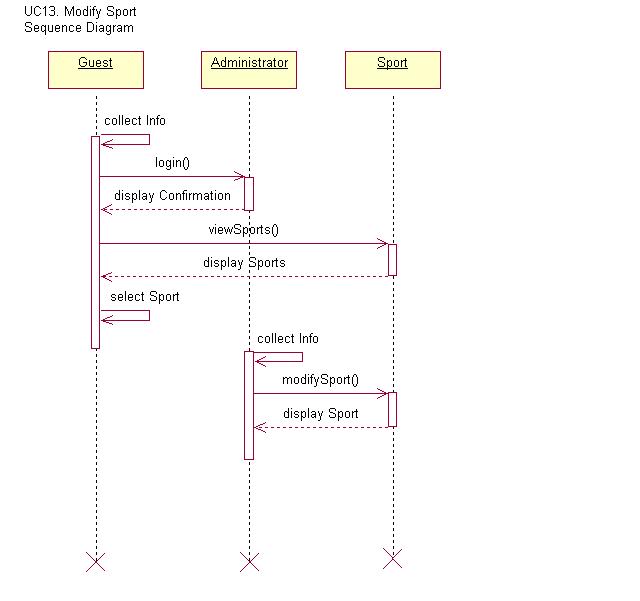
### Sequence Diagram 12 – Create Sport

The scenario of when a user creates a sport. A user must first be logged in to create a sport. After a successful login, the sport name is collected. A request is made to create the sport for the currently logged in user. A display of the confirmation of the sport creation results is returned.



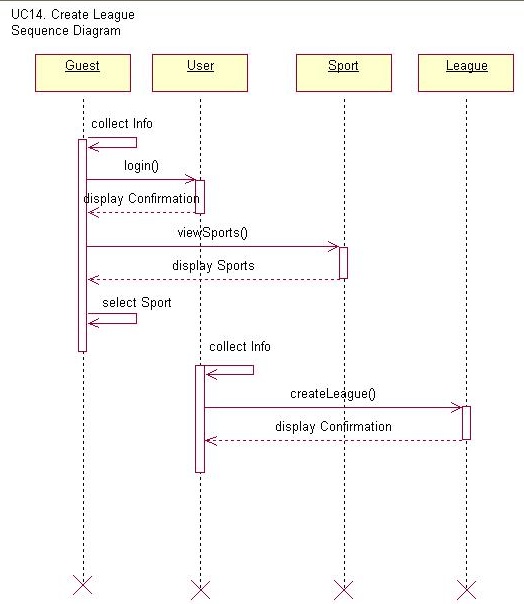
### Sequence Diagram 13 – Modify Sport

The scenario of when a user modifies a sport created by that user. A user must first be logged in to modify a sport. After a successful login, a request is made to view all available sports that they user can modify. A sport is selected from the list of sports returned. The option to update or delete a sport collected. If update has been selected, a new sport name is collected. A request is made to process the user selected option to update/delete the sport. A display of the confirmation of the modification to the sport is returned.

****

### Sequence Diagram 14 – Create League

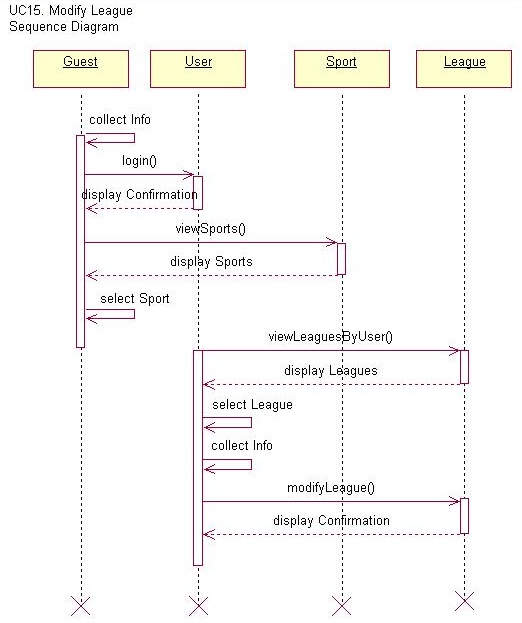
The scenario of when a user creates a league. A user must first be logged in to create a league. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A league name is collected and a request is made to create the league under the selected sport for the currently logged in user. A display of the confirmation of the league creation results is returned.

****

### Sequence Diagram 15 – Modify League

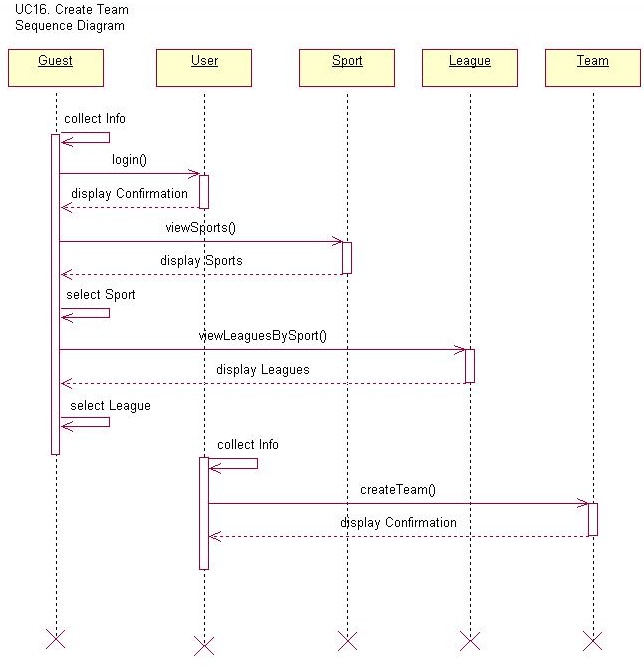
The scenario of when a user modifies a league created by that user. A user must first be logged in to modify a league. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned.

A request is made to view all leagues for the selected sport that the user can modify. A league is selected from the list of leagues returned. The option to update or delete a league is collected. If update has been selected, a new league name and sport name are collected. A request is made to process the user selected option to update/delete the league. A display of the confirmation of the modification to the league is returned.

****

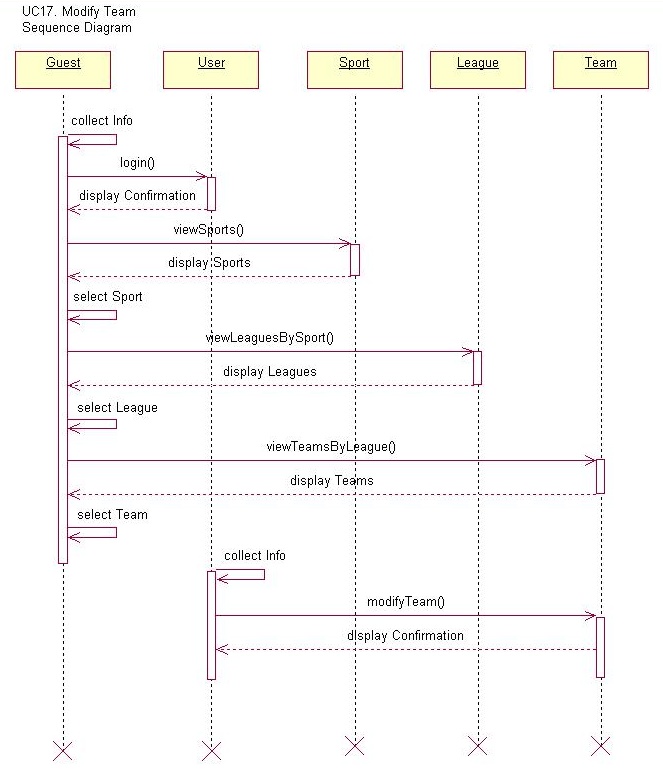
### Sequence Diagram 16 – Create Team

The scenario of when a user creates a team. A user must first be logged in to create a team. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A team name is collected and a request is made to create a team under the league, under the selected sport for the currently logged in user. A display of the confirmation of the team creation results is returned.



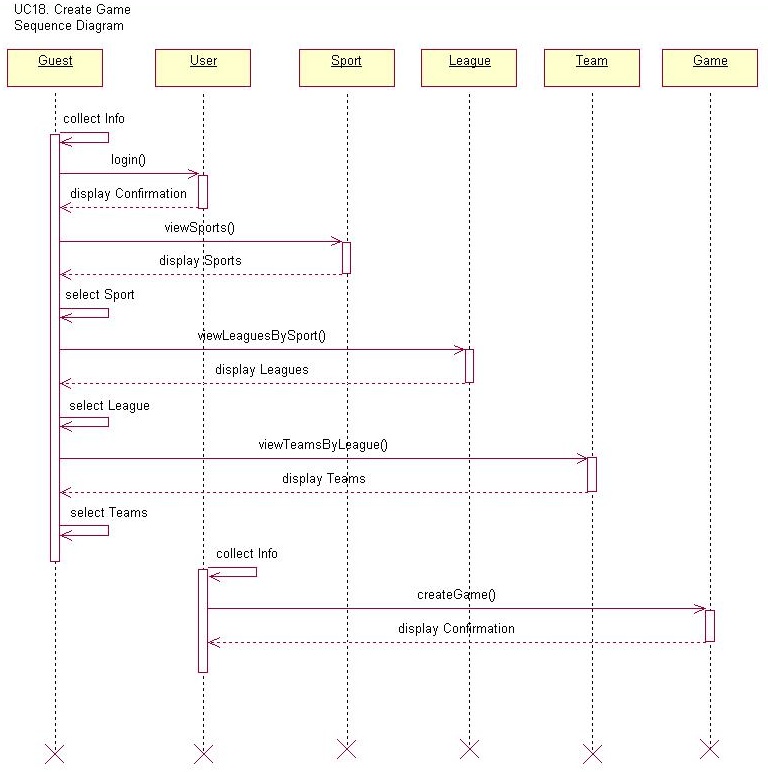
### Sequence Diagram 17 – Modify Team

The scenario of when a user modifies a team created by that user. A user must first be logged in to modify a team. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view all teams for the selected league that they user can modify. A team is selected from the list of teams returned. The option to update or delete a team is collected. If update has been selected, a new team name, league name and sport name are collected. A request is made to process the user selected option to update/delete the team. A display of the confirmation of the modification to the team is returned.



### Sequence Diagram 18 – Create Game

The scenario of when a user creates a game. A user must first be logged in to create a game. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned. A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view all teams for the selected league. Details for both teams participating in a game such as home team name, home team score, away team name and away team score and game date are collected. A request is made to create a game for the team under the league, under the selected sport for the currently logged in user. A display of the confirmation of the game creation results is returned.

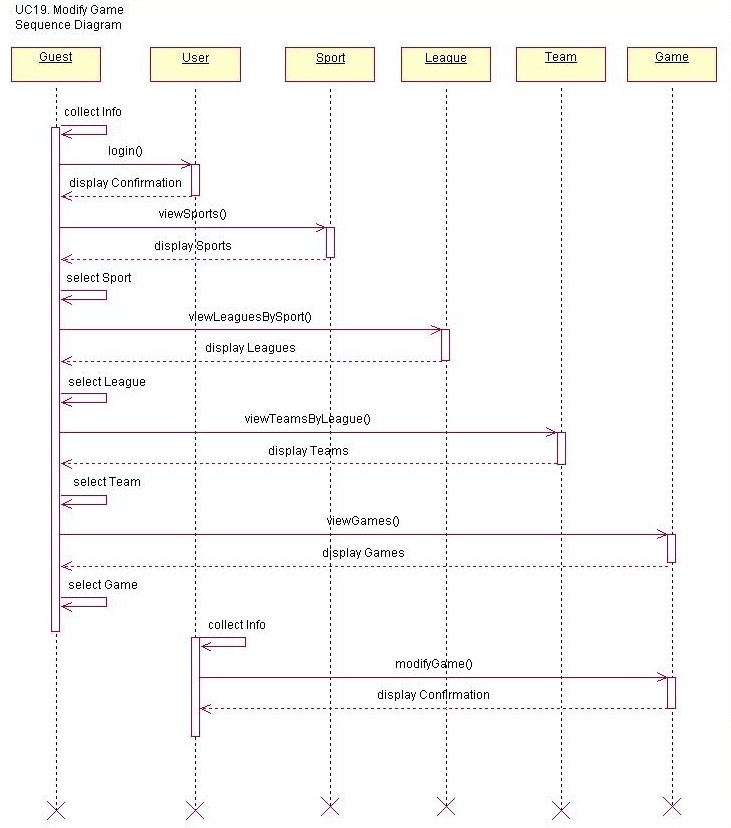


### Sequence Diagram 19 – Modify Game

The scenario of when a user modifies a game created by that user. A user must first be logged in to modify a game. After a successful login, a request is made to view all available sports. A sport is selected from the list of sports returned.

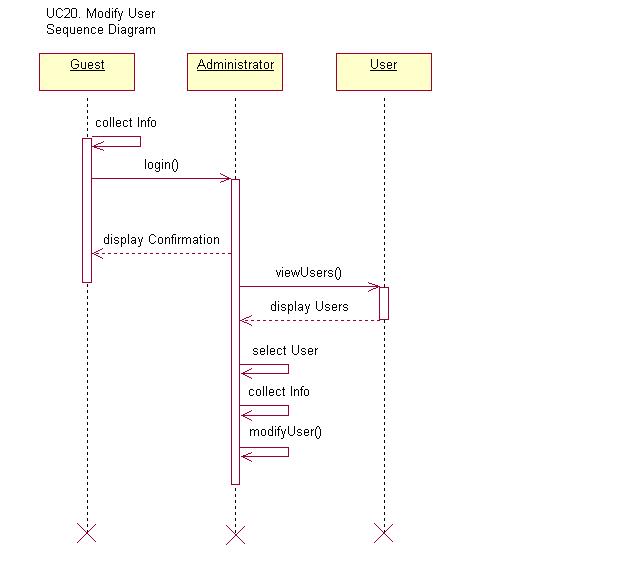
A request is made to view all leagues for the selected sport. A league is selected from the list of leagues returned. A request is made to view all teams for the selected league. A team is selected from the list of teams returned. A request is made to view all games for the selected team that the user can modify. A game is selected from the list of games returned. The option to update or delete a game is collected. If update has been selected, a new home team name, home team score, away team name, away team score and game date are collected. A request is made to process the user selected option to update/delete the game. A display of the confirmation of the modification to the game is returned.

(Shown on next page)



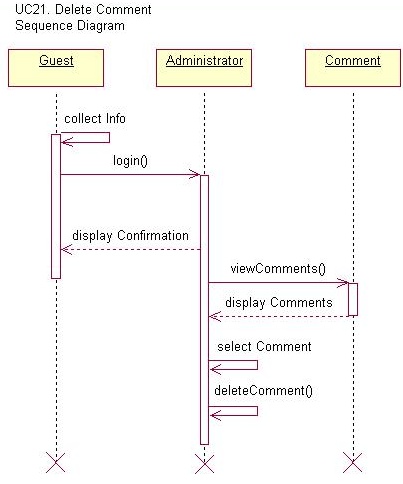
### Sequence Diagram 20 – Modify User

The scenario of when an administrator chooses to modify a user. An administrator must first be logged in to modify a user. A request is made to view all users in the system. A user is selected from the list of users returned. The option to update or delete a user is collected. If update has been selected, a new user email is collected. A request is made to process the administrator selected option to update/delete the user.

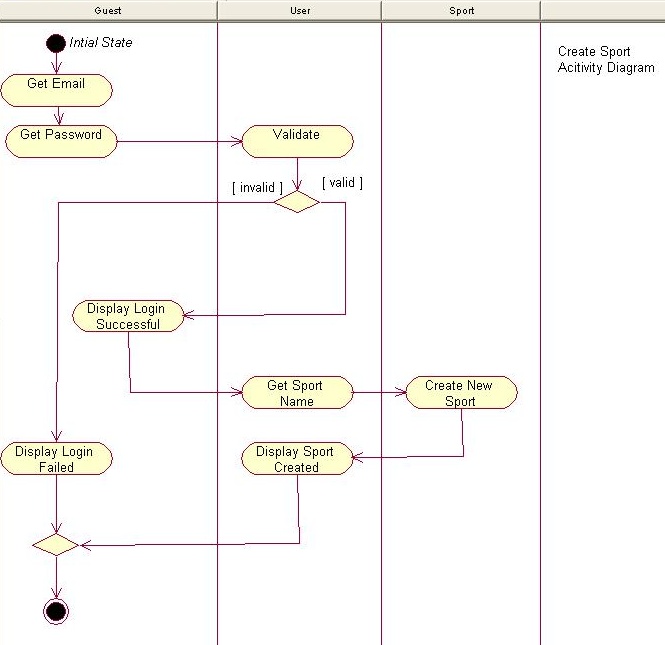


### Sequence Diagram 21 – Delete Comment

The scenario of when an administrator chooses to delete a comment posted by a user. An administrator must first be logged in to delete comments. A request is made to view all comments in the system. A comment is selected from the list of comments returned. The option to delete a comment is collected and a request is made to process the deletion of the selected comment.



# Activity Diagrams





# Database Diagrams

## Database Tables



## Database ER Model



# Conclusion

In conclusion, The Sports Score Tracker fulfilled all of the requirements of the problem posed by the Southeastern Cal Ripken Baseball League.

Instead of having to hand enter and update all of the league data, Southeastern officials can now track all league and team information on the Internet in the Sports Score Tracker. The system handled all of the leagues and teams within Southeastern Cal Ripken, as well as all of the leagues and teams that work hand-in-hand with Southeastern.

Furthermore, the Sports Score Tracker allows for the addition of other sports that can also be tracked via the World Wide Web.

# ­Data Dictionary

### Class: Guest

The Guest object is used to store attributes and functions related to a guest user that is not logged in.

**Attributes:**

**Functions:**

resetPassword(): This function will email the user a new randomly selected password.

email = GET email address

user = retreiveUserFromRecords(email)

if (user not found)

newPassword = getRandomPassword()

modifyCurrentUserPassword(newPassword)

SendEmail(email, newPassword)

return true

else

return false

checkEmail(): This function checks if an email is already registered in the system, returns true if email does not exist.

email = GET email address

user = retreiveUserFromRecords(email)

if (user not found)

return true

else

return false

### Class: User

The User object is used to store attributes and functions related to a single user in the system.

**Attributes:**

email::string: This is the unique email address of the user, it will be used for login and

firstName::string: This is the first name of the user.

ID:: int: This is the unique ID of the user.

isAdmin::bool: This is a Boolean value specifying rather this user is an admin or not.

lastName::string: This is the last name of the user.

password::string: This is the password of the user; it will be stored as an MD5 hash of the actual password.

**Functions:**

login(): This will check the email and password for a user to verify they match a user in the system, and log the user in.

email = GET email address

password = GET password

passwordHash = MD5(password)

user = retreiveUserFromRecords(email, passwordHash)

if (user not found)

return user.ID

else

return -1

logout(): This will end the current user’s session.

Session.Destroy()

redirect to welcome page

modifyPassword(): This will confirm the user’s current password, and if it matches update the password to a new password the user enters.

oldPass = GET old password

newPass = GET new password

if (oldPass matches old password)

modifyCurrentUserPassword(MD5(newPass))

return true

else

return false

registerUser(): This will register and save the new user’s information for a guest.

user = new User()

user.email = GET email address

user.firstName = GET first name

user.lastName = GET last name

user.password = MD5(GET password)

SendEmail(user.email, confirmation text)

### Class: Game

The Game object is used to store attributes and functions related to a Game. It is what defines a game, such as the home team, away team, each team’s score and the date/time the game is played.

**Attributes:**

awayScore::int: This is the away team’s score for a game.

awayTeamID:: int: This is the ID of the away team.

date::datetime: This is the date/time the game is played.

homeScore:: int: This is the home team’s score for a game.

homeTeamID:: int: This is the ID of the home team.

ID:: int: This is the unique ID for each game.

**Functions:**

createGame(): This function creates a new game.

game = new Game()

game.homeTeamID = GET home team ID

game.awayTeamID = GET away team ID

game.homeScore = GET home score

game.awayScore = GET away score

game.date = GET game date

game.Save()

viewGames(): This function will allow viewing all games that a particular team participates in.

teamID = get team ID to view games for

return retreiveGamesFromRecords(teamID)

modifyGame(): This function will allow updating and deleting of a game in the system.

gameID = GET game ID

game = GetGameByID(gameID)

if (user option == updating)

game.homeTeamID = GET new home team ID

game.awayTeamID = GET new away team ID

game.homeScore = GET new home score

game.awayScore = GET new away score

game.date = GET new game date

game.Save()

else

game.Delete()

viewScheduleByLeague(): This will view the game schedule for all teams in a league. The schedule will be represented as the date/time of the game, and the names of the teams playing the game.

leagueID = GET selected league

return retreiveScheduleByLeague(leagueID)

postComment(): This will allow a user to post a comment into the system for a specific game, of a certain type.

game = GetGameByID(gameID)  
game.comment = GET comment  
comment.timestamp = GET timestamp  
comment.type = GET type  
comment.Save()

publishVote(): This will publish a user’s vote for which team they think will win a game.

prediction = new Prediction()

prediction.gameID = GET game ID

prediction.teamID = GET team ID

prediction.useriD = GET user ID

prediction.save()

viewComments(): This will view all the comments for a certain game.

gameID = GET ID of game to view comments for

return retreiveCommentsForGame(gameID)

viewPredictions(): This will view the system estimated and user estimated prediction for a single game. The system estimated prediction will be presented just as a name of the team that will win, while the user predictions will be summed up by which team is predicted to win, and represented as a percentage of the voting results.

gameID = GET game ID

return retreivePredictionsForGame(gameID)

viewScore(): This will view the score for a game. It will show the name of each team, and each team’s score.

gameID = GET gameID

return retrieveScoreForGame(gameID)

### Class: Predictions

The Predictions object is used to store attributes and functions related to a user’s prediction as to which team will when a game.

**Attributes:**

gameID:: int: This is the ID of the game the user is making a prediction for.

teamID:: int: This is the ID of the team the user predicts to win.

userID:: int: This is the ID of the user making the prediction.

### Class: Comment

The Comment object is used to store attributes and functions related to a comment placed by a user for a given game.

**Attributes:**

comment::string: This is the text of the user’s comment.

gameID:: int: This is the ID of the game the user is commenting on.

ID:: int: This is the unique ID of the comment.

timestamp::datetime: This is the date/time the comment was placed into our system.

Type::short: This is the type of comment the user placed.

userID::int: This is the ID of the user who created the comment.

**Functions:**

### Class: Team

The Team object is used to store attributes and functions related to a single team in the system.

**Attributes:**

ID::int: This is the unique ID of the team.

leagueId::int: This is the ID of the league this team is a member of.

teamName::string: This is the name of the team.

**Functions:**

viewTeamsByLeague(): This will view all the teams in a league.

leagueID = GET league ID

return retreiveTeamsByLeague(leagueID)

createTeam(): This will create a new team.

team = new Team()

team.leagueID = GET league ID

team.name = GET team name

team.Save()

modifyTeam(): This function will allow updating and deleting of a team currently in the system.

teamID = GET team ID

team = retreiveTeamByID(teamID)

if (user option == updating)

team.league = GET league ID

team.name = GET team name

team.Save()

else

team.Delete()

### Class: Administrator

The Administrator object is used to store attributes and functions related to a single administrator in the system. This object will inherit from the base User object.

**Attributes:**

**Functions:**

modifyUser(): This function will allow updating and deleting of users currently registered in the system.

userID = GET user ID

user = retreiveUserFromRecords(userID)

if (user option == updating)

user.email = GET email address

user.firstName = GET first name

user.lastName = GET last name

user.password = MD5(GET password)

else

user.Delete()

deleteComment(): This function will allow the administrator to delete comments placed by users in the system. This is used if a derogatory or otherwise invalid comment is posted and the administrator wishes to delete it.

commentID = GET comment ID  
comment = getCommentByID(commentID)  
comment.Delete()

### Class: League

The League object is used to store attributes and functions related to a single League in the system.

**Attributes:**

ID::int: This is the unique ID of the league.

Name::string: This is the name of the league.

sportID::int: This is the ID of the sport type which this league belongs to.

userID::int: This is the ID of the owner of this league. The user with this ID will be the only one to be able to modify this league.

**Functions:**

createLeague(): This function will create a new league.

league = new League()

league.name = GET league name

league.sportId = GET league sportId

league.userId = GET current user Id

league.Save()

viewLeaguesBySport(): This function will display all the leagues that belong to a sport.

sportID = GET sport ID

return retreiveLeaguesBySport(sportID)

viewLeaguesByUser(): This function will display all the leagues that are owned by a user.

userID = GET current user ID  
return retreiveLeaguesByUser(userID)

modifyLeagues(): This function will allow updating and deleting of a league.

leagueID = GET league ID

league = GetLeagueByID(leagueID)

if (user option == updating)

league.name = GET new league name

league.sportId = GET new league sportId

league.userId = GET new league userId

league.Save()

else

league.Delete()

### Class: Sport

The Sport object is used to store attributes and functions related to a single sport in the system.

**Attributes:**

ID::int: This is the unique ID of the sport type.

Name::string: This is the name of the sport type, for example: Basketball or Football.

**Functions:**

viewSports(): This function will show all the sports that are in the system.

return retreiveSports()

createSport(): This function will allow creation of a new sport.

sport = new Sport()  
sport.Name = GET sport Name  
sport.Save()

modifySport(): This function will allow the administrator to update or delete a sport.

sportID = GET sport ID

sport = GetSportByID(sportID)

if (user option == updating)

sport.name = get new sport name

sport.Save()

else

sport.Delete()