



Iteration 2 - March 23, 2018

By Group 4:
Elizabeth Beisert
Kevin Dang
Daniel Hussey
Roman Soriano
Gerald Zapata




Table of Contents

| | |
|---------------------------------------|-----------|
| Note To Grader | 3 |
| Terminology | 4 |
| Requirements | 5 |
| Use Cases | 7 |
| Abstract Use Cases | 7 |
| Requirements - Use Case Matrix | 9 |
| Expanded Use Cases | 10 |
| Use Case Diagram | 15 |
| Domain Model | 17 |
| Brainstorming / Classification | 18 |
| Scenarios / Tables | 19 |
| Sequence Diagram | 24 |
| Design Class Diagram | 26 |
| Software Demo | 27 |

Note To Grader

For this iteration, all use cases are delivered, but only:

- UC1: Search for LCSPlayer
- UC2: Compare LCSPlayer
- UC4: Create User
- UC10: View Matchup
- UC5: Create League

Are developed through the scenario/tables, sequence diagram, and thus reflected in the design class diagram

Deliverables

For Iteration 1 we provided a **search function** to search up LCSPlayers in the database

For Iteration 2 we provided a **login GUI that allows creation of accounts and logging in**. We also added tabs to the main UserInterface to allow **LCSPlayer comparisons, League creations, and viewing of UserTeam matchup scores** each week.

Iteration Planning

For each iteration, the project **documentation** and **planning** will exist first before any programming is done to add the new features. We will also discuss in a group how we want classes and program flow to be set up before we begin programming. If meeting a requirement is unfeasible for a particular iteration, the deliverables will be modified and plans be made to fix the issues that stopped production. Project documentation and planning should be finished 4 days before the deadline and coding begin immediately.

Terminology

Team- A team of players put together by a User for fantasy League of Legends

League- A group of 4, 6, or 8 Users who each have a team and compete for the statistically best team

User- A person using Observer software to play fantasy League of Legends, responsible for managing Teams

Admin- Administrators or a person managing the software for Observer, responsible for creating Teams and Leagues and associating the corresponding Users to them

LCSPlayer- A person who plays in the League of Legends Championship Series

LCSTeam- A team of LCSPlayers that plays in the League of Legends Championship Series

Free Pool- LCSPlayers that do not belong to a Team within a League

Matchup - Two teams are are current playing against each other in a specific week

Swap- Switching an LCSPlayer in a Team with one from the Free Pool

Trade- Switching an LCSPlayer in a Team with one from another Team in the league; Requires approval from the Users managing each Team

Role- The role an LCSPlayer holds on their LCSTeam

Starting- This player's points will count toward the total score for a matchup.

There are 7 on each Team, one for each Role, and one Flex spot, which can hold any LCSPlayer

Bench - These players belong to one User's Team, but do not count toward the matchup score. There are 3 Bench spots on each team

Abbreviations:

UC- Use Case

R- Requirement

Requirements

Actors

User

Admin

Scope of Project

Observer is a local software that designates the first person to use it as the Admin and all subsequent users as Users. There will be only instance of a Observer software for each “grouping” of people who are playing fantasy League of Legends together. Observer is planning to use internet in order to pull League of Legend professional player information from the official League of Legend championship website, but as of now pulls data locally as supplied by the users of the software.

Description of Project

Fantasy football is the most famous of the fantasy league games, but fantasy League of Legends also has a big following, although there is little support for players. Observer will calculate statistics for the fantasy teams and allow comparison of these fantasy teams within fantasy leagues, like in many fantasy football applications.

Requirements

| Priority Weight (1 - 5) | Requirements |
|-------------------------|--|
| 3 | R1. The system (SUD) shall generate database of LCSPlayers by reading a spreadsheet document and placing relevant information into a database class |
| 2 | R2. The system shall display all LCSPlayers in app by sorting according to any of the available data fields (excluding flags). <i>For reference:</i> http://fantasy.na.lolesports.com/en-US/league/1231363/stats |
| 2 | R3. The system shall allow for selective viewing of data based on specific fields or keywords. 3.1 User will search for LCSPlayers by selecting a specific Role |
| 2 | R4. The system should create a search function which allows user to search for a player or LCS team specifically. 4.1 User will search for a player or LCS team by typing keywords |
| 1 | R5. <i>If possible:</i> The system shall allow for comparison of two LCSPlayers by quantitative data (IE points, kills, deaths). 5.1 The User will select two LCSPlayers to compare. |
| 4 | R6. The system shall create a User. 6.1 User will select “Create Account” and enter the information required in the subsequent page |

| | |
|---|--|
| 4 | <p>R7. The system shall create a League.</p> <p>7.1 The Admin will select to create a League and choose Users to play in it</p> |
| 4 | <p>R8. The system shall create a Team within a League.</p> <p>8.1 The Admin will select to create a Team once a League is created and assign a User to it</p> |
| 4 | <p>R9. The system shall drop an LCSPlayer from a Team within a League.</p> <p>9.1 The User will select an LCSPlayer and choose the option to drop the LCSPlayer</p> |
| 4 | <p>R10. The system shall add an LCSPlayer to a Team within a League.</p> <p>10.1 The User will select an LCSPlayer from the Free Pool and choose to add the LCSPlayer to the User's Team</p> |
| 5 | <p>R11. The system shall calculate and display Team statistics on log in.</p> |
| 4 | <p>R12. The system shall allow a User to modify their Team's starting LCSPlayers</p> <p>12.1 The User will select a LCSPlayer in a Starting slot and a player in a Bench slot to exchange places</p> |

Constraints:

1. The Observer software must be operated by one person at a time
2. The LCSPlayer data cannot be altered in any way except on creation

Abstract Use Cases

User Use Cases

- UC1: Search for LCSPlayer
- UC2: Compare LCSPlayer
- UC3: Swap LCSPlayer
- UC4: Create User
- UC6: Propose Trade
- UC7: Accept Trade
- UC9: Change Starting Roster
- UC10: View Matchup

Admin Use Cases

- UC5: Create League
- UC8: Create a Team

High Level Use Cases

High Level User Use Cases:

- **UC1 Search for LCSPlayer:**
 - TUCBW the User opening the search tab
 - TUCEW the User sees the desired LCSPlayer or LCSTeam
- **UC2 Compare LCSPlayer:**
 - TUCBW the User opening the Compare tab
 - TUCEW the User sees the two LCSPlayers compared
- **UC3 Swap LCSPlayer:**
 - TUCBW the User opening the Swap tab
 - TUCEW the User confirms the swap and returns to their roster screen
- **UC4 Create User:**
 - TUCBW the User opens the “Create Account” tab
 - TUCEW the User receives confirmation that an account has been created and goes to user homepage
- **UC6 Propose Trade:**
 - TUCBW User 1 selects the Initiate Trade option
 - TUCEW User 1 confirming trade message to User 2
- **UC7 Accept Trade:**
 - TUCBW User opening the trade request
 - TUCEW User confirming or denying the trade request
- **UC9 Change Starting Roster:**
 - TUCBW User opening the Edit Roster menu
 - TUCEW User confirming new roster submission
- **UC10 View Matchup:**
 - TUCBW User selects the Matchups tab
 - TUCEW User views the selected matchup

High Level Admin Use Cases:

- **UC5 Create League:**
 - TUCBW the Admin selects “Create League” tab
 - TUCEW the Admin returns to current existing leagues and sees their created league added to list
- **UC8 Create a Team:**
 - TUCBW Admin selecting Create Team from the league creation menu and entering team name
 - TUCEW Admin confirms the user addition to the team

Requirements - Use Case Traceability Matrix

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | Score |
|----------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-------|
| Priority | 3 | 2 | 2 | 2 | 1 | 4 | 4 | 4 | 4 | 4 | 5 | 4 | |
| UC1 | | x | x | x | | | | | | | | | 6 |
| UC2 | | | | | x | | | | | | | | 1 |
| UC3 | | | | | | | | | x | x | | | 8 |
| UC4 | | | | | | x | | | | | | | 4 |
| UC5 | | | | | | | x | | | | | | 4 |
| UC6 | | | | | | | | | x | x | | | 8 |
| UC7 | | | | | | | | | x | x | | | 8 |
| UC8 | | | | | | | | x | | | | | 4 |
| UC9 | | | | | | | | | | | | x | 5 |
| UC10 | | | | | | | | | | | x | | 4 |

Requirement 1 is a background process that will use an event driven architecture. R11 will be available with each league, but is not a part of a business process.

Expanded Use Cases

Use Case 1: Search for LCSPlayer

| | |
|---|--|
| Actor: User | System: Observer |
| | 0) System displays database menu screen |
| 1) User opens the search tab | 2) System displays the search bar and results list |
| 3) User enters LCSPlayer or LCSTeam name | 4) System returns shortlist of LCSPlayer names matching criteria |
| 5) User selects name of desired LCSPlayer | 6) System fetches complete data for selected LCSPlayer |
| 7) User sees desired LCSPlayer or LCSTeam | |

Use Case 2: Compare LCSPlayer

| | |
|---|---|
| Actor: User | System: Observer |
| | 0) System displays database menu screen |
| 1) User opens compare tab | 2) System displays dual search boxes on the compare screen |
| 3) User enters LCSPlayer or LCSTeam names | 4) System returns shortlist of LCSPlayer names matching criteria |
| 5) User selects name of desired LCSPlayer from each box | 6) System fetches complete data for selected LCSPlayer and calculates comparison statistics |
| 7) User sees the two LCSPlayers compared | |

Use Case 3: Swap LCSPlayer

| | |
|---|--|
| Actor: User | System: Observer |
| | 0) System displays free agents roster |
| 1) User opens swap tab | 2) System displays list of available free agents |
| 3) User selects a LCSPlayer from the free agent list | 4) System prompts user to select a LCSPlayer from their roster to drop |
| 5) User selects LCSPlayer they wish to exchange | 6) System asks for confirmation on add/drop |
| 7) User confirms LCSPlayer swap and returns to their roster | |

Use Case 4: Create User

| | |
|--|--|
| Actor: User | System: Observer |
| | 0) System displays menu screen |
| 1) User opens "Create Account" tab | 2) System displays new page filled with blank boxes for user to fill |
| 3) User enters personal information | 4) System checks if account with current email or username exists |
| 5) User receives confirmation that an account has been created and goes to user homepage | |

Use Case 5: Create League

| | |
|--|---|
| Actor: User | System: Observer |
| | 0) System displays current existing leagues |
| 1) Admin selects “Create League” tab | 2) System displays window with empty boxes for user to assign league name and number of Teams |
| 3) Admin enters League name and number of Teams (limited to 4, 6, or 8) within league | 4) System checks if any league of the entered name exist |
| 5) Admin confirms the League name and Team number | 6) System returns the league has been created and adds it to the database |
| 7) Admin returns to current existing leagues and sees their created league added to list | |

Use Case 6: Propose Trade

| | |
|--|---|
| Actor: User | System: Observer |
| | 0) System displays another Team’s roster screen |
| 1) User 1 selects the Initiate Trade option | 2) System prompts the user to select one of the User 2’s roster LCSPlayers |
| 3) User 1 selects a LCSPlayer from User 2’s Team | 4) System prompts User 1 to select one of their own roster LCSPlayers |
| 5) User 1 selects a LCSPlayer to send to User 2 | 6) System verifies both rosters will be legal after the swap and a. Prompts User1 to attach a message to the trade request b. Prompts User1 to select another LCSPlayer, returns to step 3) |
| 7) User 1 enters the desired message | 8) System shows confirmation message with both LCSPlayers being traded and the message |
| 9) User 1 confirms trade message to User 2 | |

Use Case 7: Accept Trade

| | |
|---|---|
| Actor: User | System: Observer |
| | 0) The system displays a notification saying that a trade request has been received |
| 1) User opens the trade request | 2) System displays the Trade Summary |
| 3) User a. Selects More Details option b. Ignores More Details option, moves to 5 | 4) a. System displays comparison tab for both players, with an option to return to the trade |
| 5) User a. Confirms trade request b. Denies trade request | |

Use Case 8: Create a Team

| | |
|---|--|
| Actor: Admin | System: Observer |
| | 0) System displays the Team Creation options |
| 1) Admin selects Create Team from the menu and enters team name | 2) System allows Admin to add user to selected Team |
| 3) Admin enters the account name of the User | 4) System verifies the existence of the User and adds User to the Team |
| 5) Admin confirms the user addition to the team | |

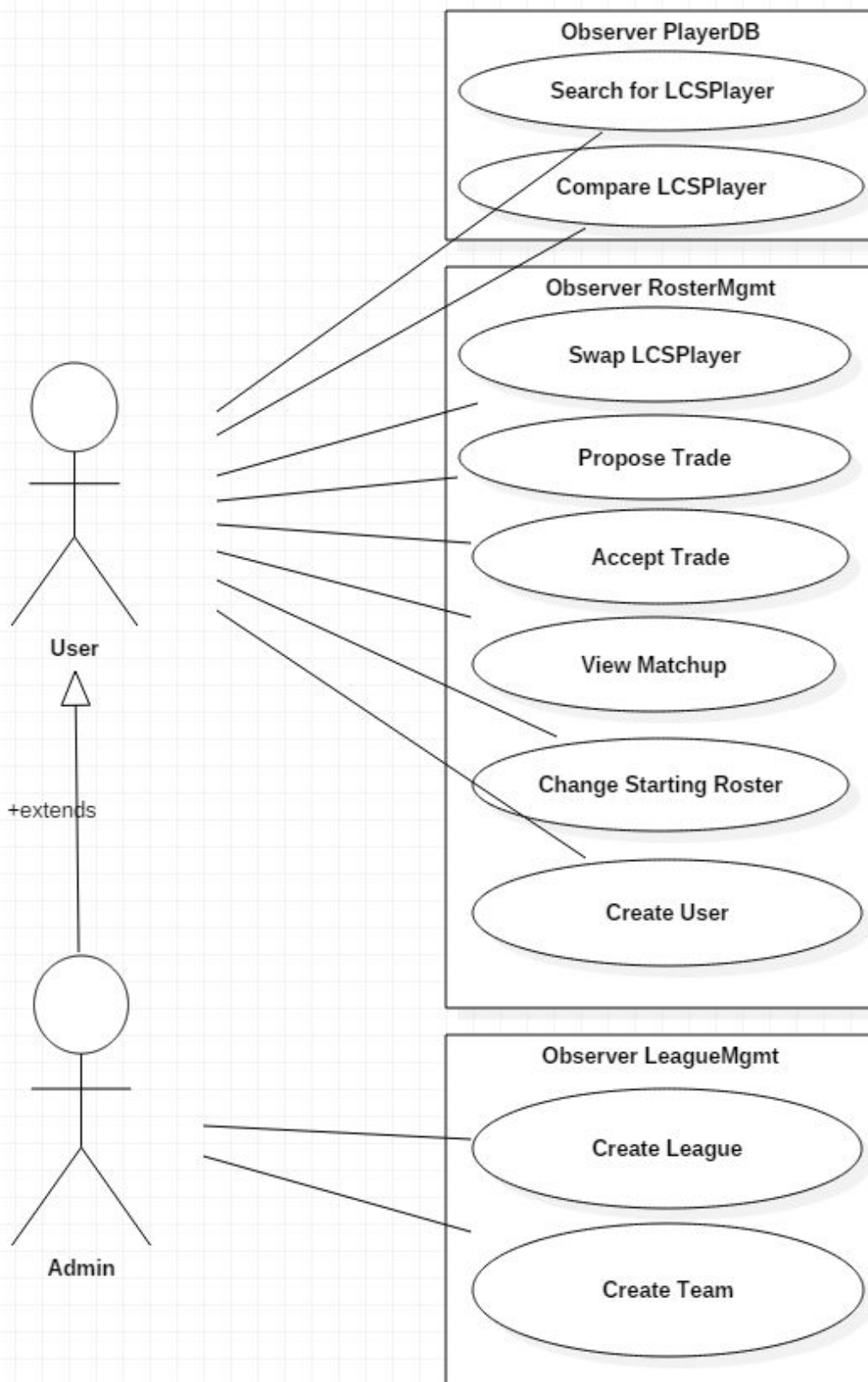
Use Case 9: Change Starting Roster

| | |
|--|---|
| Actor: User | System: Observer |
| | 0) System displays Roster screen |
| 1) User opens the Edit Roster Menu | 2) System prompts User to select two players to exchange places |
| 3) User selects which players to move | 4) System verifies that the move is legal, performs exchange, and prompts User for confirmation |
| 5) User confirms the roster submission | |

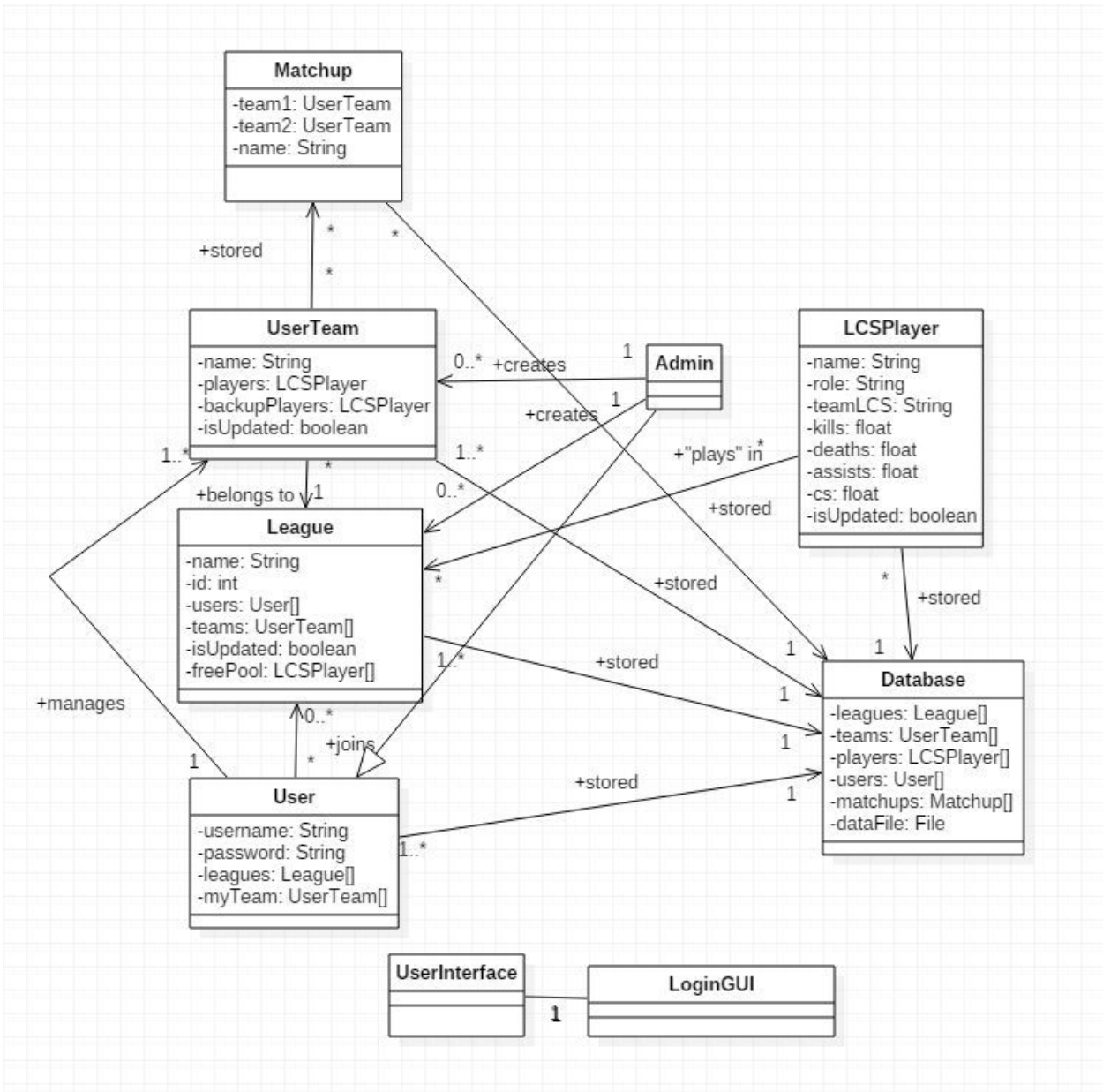
Use Case 10: View Matchup

| | |
|--|--|
| Actor: User | System: Observer |
| | 0) System displays the home screen |
| 1) User selects the Matchups tab | 2) System retrieves and displays the names of the teams in the week's matchup |
| 3) User a) Chooses a matchup from the list b) Chooses another week to view | 4) System a) Retrieves detailed data for each team and displays full score screen b) System retrieves the chosen week's matchups. Return to 3) |
| 5) User views the selected matchup | |

Use Case Diagram



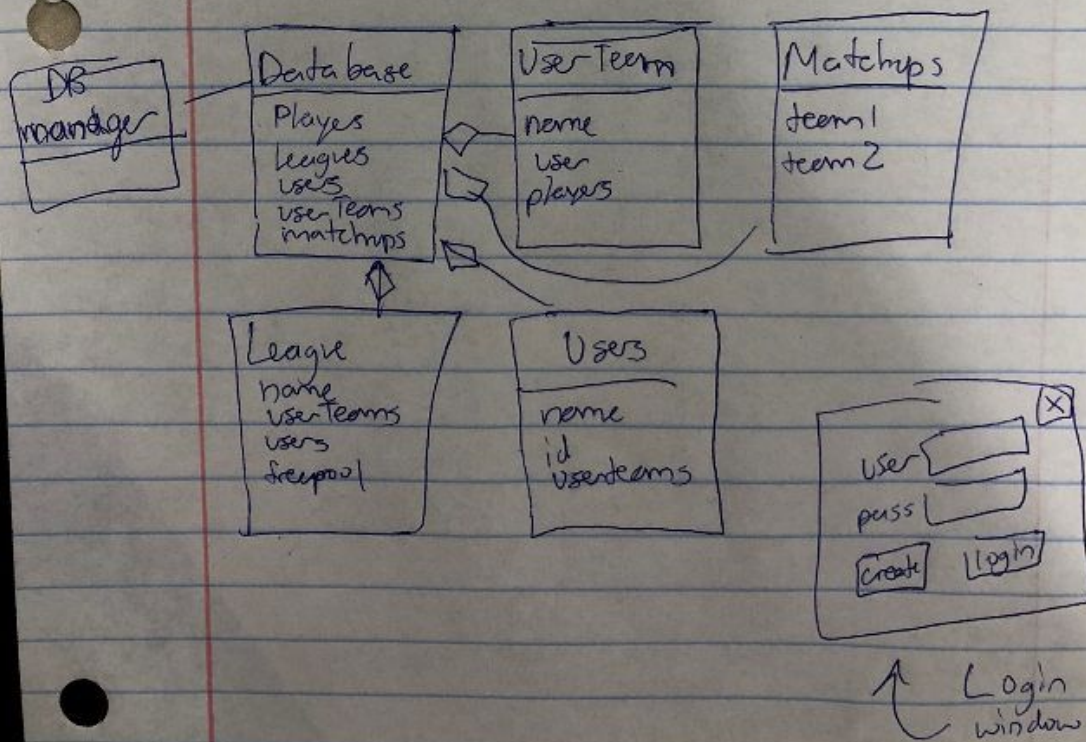
Domain Model



Brainstorming / Classification

- (inherit)
- Classification
1. User \rightarrow Admin
 2. LCSPlayer
 3. User's team of LCSplayers
 4. Leagues of User's teams
 5. Matchups between teams
 6. Database (and manager)

3. User manages which LCSplayers they have
4. Leagues are created by Admin
6. All data aggregations stored in Database



Scenarios / Tables

Scenario 1: UC 1 Database searching

1. User selects a Role or LCSTeam dropdown menu and selects a criteria option.
2. UserInterface calls the Search Controller and passes it the search criteria.
3. SearchController calls DBManager.
 - 3.1. The DBManager retrieves a list of all LCSPlayers.
4. The SearchController creates an empty list.
 - 4.1. For each player,
 - 4.1.1. The SearchController passes the LCSPlayer the search criteria to verify if it matches them.
 - 4.1.2. If the LCSPlayer matches with the search criteria.
 - 4.1.3. The SearchController adds the player to the list.
5. The SearchController returns the list of LCSPlayers to the UserInterface.
6. UserInterface displays the list of LCSPlayers that match the criteria.

| | Subject | Action of Subject | Other Data/Objects | Object Acted Upon |
|-------|------------------|-------------------|--------------------|----------------------|
| 1 | UserInterface | selects | criteria | Dropdown menu option |
| 2 | UserInterface | calls | | Search Controller |
| 3 | SearchController | calls | | DBManager |
| 3.1 | DBManager | retrieves | | List of LCSPlayers |
| 4 | SearchController | creates | | Empty list |
| 4.1 | | | | |
| 4.1.1 | SearchController | passes | LCSPlayer | SearchController |
| 4.1.2 | | | | |
| 4.1.3 | SearchController | adds | LCSPlayer | List |
| 5 | SearchController | returns | List of LCSPlayers | UserInterface |
| 6 | UserInterface | displays | | List of LCSPlayers |

Scenario 2: UC 2 Comparing LCSPlayer

1. UserInterface's Compare tab shows two parallel dropdown search menus.
2. User selects a Role or LCSTeam dropdown menu and selects a criteria option for the first LCSPlayer.
3. Compare GUI calls the Search Controller and follows Scenario 1.
4. The Search Controller returns the list of LCSPlayers to the Compare GUI.
5. Search GUI displays the list of LCSPlayers that match the criteria.
6. User selects the LCSPlayer searched for.
7. The LCSPlayer's information is shown on half of the screen.
8. User selects a Role or LCSTeam dropdown menu and selects a criteria option for the second LCSPlayer.
9. Search GUI calls the Search Controller and follows Scenario 1.
10. The Search Controller returns the list of LCSPlayers to the Compare GUI.
11. Search GUI displays the list of LCSPlayers that match the criteria.
12. User selects the LCSPlayer searched for.
13. The LCSPlayer's information is shown on the other half of the screen.

| | Subject | Action of Subject | Other Data/Objects | Object Acted Upon |
|----|------------------|-------------------|--------------------|----------------------|
| 1 | UserInterface | displays | | Drop Down |
| 2 | UserInterface | selects | criteria | Dropdown menu option |
| 3 | UserInterface | calls | | SearchController |
| 4 | SearchController | returns | List of LCSPlayers | UserInterface |
| 5 | UserInterface | displays | | List of LCSPlayers |
| 6 | UserInterface | selects | | LCSPlayer |
| 7 | UserInterface | displays | | LCSPlayer |
| 8 | UserInterface | selects | criteria | Dropdown menu option |
| 9 | UserInterface | calls | | SearchController |
| 10 | SearchController | returns | List of LCSPlayers | UserInterface |
| 11 | UserInterface | displays | | List of LCSPlayers |
| 12 | User | selects | | LCSPlayer |
| 13 | UserInterface | displays | | LCSPlayer Stats |

Scenario 4: UC 4 Creating User

1. User selects the option to Create Account
2. Login GUI displays account creation form
3. User enters their information to create the account
4. Login GUI calls the Account Controller and passes in the User information
5. Account Controller Calls DBManager
 - 5.1. DBManager checks the database for duplicates
6. Account Controller creates a new User entry with the information
 - 6.1 Account Controller passes the new User to the DBManager
 - 6.1.1 DBMgr adds the new User to the database
7. Login GUI displays a “creation successful” message
8. Login GUI displays the User homepage

| | Subject | Action of Subject | Other Data/Objects | Object Acted Upon |
|-------|--------------------|-------------------|--------------------|-----------------------|
| 1 | Login GUI | selects | | Creation menu |
| 2 | Login GUI | displays | | Account creation form |
| 3 | Login GUI | receives | | User information |
| 4 | Login GUI | calls | | Account Controller |
| 5 | Account Controller | passes | User information | DBMgr |
| 5.1 | DBManager | checks | duplicate entries | List |
| 6 | Account Controller | creates | | User |
| 6.1 | Account Controller | passses | User | DBManager |
| 6.1.1 | DBManager | adds | User | Database |
| 7 | Login GUI | displays | | confirmation |
| 8 | Login GUI | displays | | UserInterface |

Scenario 5: UC 5 Creating League

1. Admin selects option to Create League
2. UserInterface displays League creation form
3. Admin enters League name and number of LCSTeam's that will participate in new league
4. UserInterface calls the League controller
5. LeagueController calls the DBManager
 - 5.1. DBManager retrieves the list of current existing leagues and checks that new league does not currently exist.
6. LeagueController returns confirmation that League has been created.
7. UserInterface displays league that Admin created

| | Subject | Action of Subject | Other Data/Objects | Object Acted Upon |
|-----|------------------|-------------------|--------------------------------|--------------------------|
| 1 | UserInterface | selects | | League tab |
| 2 | UserInterface | display | | Drop Down |
| 3 | UserInterface | selects | criteria | Drop Down Menu option |
| 4 | UserInterface | Calls | | Create League Controller |
| 5 | LeagueController | calls | | DBManager |
| 5.1 | DBManager | retrieves | List of current Leagues | UserInterface |
| 6 | LeagueController | returns | Confirmation of created League | UserInterface |
| 7 | UserInterface | displays | | Created League in list |

Scenario 10: UC 10 Viewing Matchup

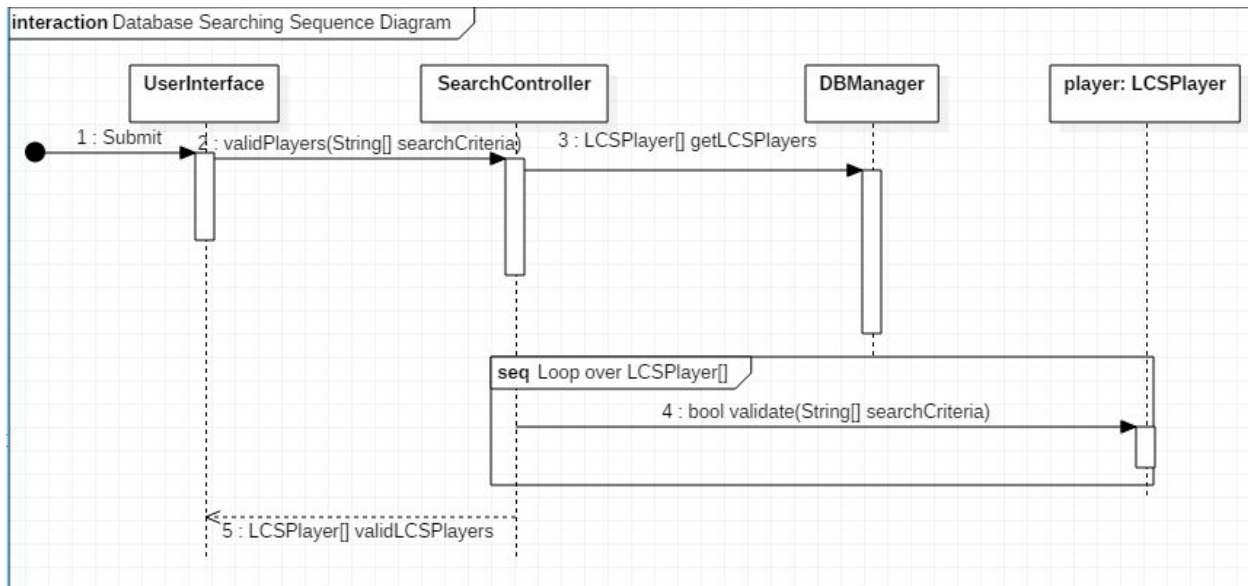
1. UserInterface calls the MatchupController
2. MatchupController calls the DBManager
 - 2.1. DBManager retrieves the list of list of Matchups
3. UserInterface displays the first index of list of Matchup objects
4. If the User selects a different week
 - 4.1. UserInterface displays the specified index of list of Matchup objects
 - 4.2. Jump to 5
5. If the User selects a Matchup from the list
 - 5.1. UserInterface displays the statistics for each UserTeam
 - 5.2. For every LCSPlayer in the Matchup on each team,
 - 5.2.1. Matchup object adds LCSPlayer attributes to sum
 - 5.3. Matchup object returns the list of two sums to Matchup GUI
 - 5.4. UserInterface displays the sum for each UserTeam

| | Subject | Action of Subject | Other Data/Objects | Object Acted Upon |
|-------|---|-------------------|----------------------|--|
| 1 | UserInterface | calls | | Matchup Controller |
| 2 | UserInterface | calls | | DBManager |
| 2.1 | DBManager | retrieves | | List of List of Matchups |
| 3 | UserInterface | display | | First week of list of Matchups |
| 4 | If the User selects a different week | | | |
| 4.1 | UserInterface | displays | | Specified index of list of Matchup objects |
| 4.2 | Jump to 4 | | | |
| 5 | If the User selects a Matchup from the list | | | |
| 5.1 | UserInterface | displays | | Statistics for each UserTeam |
| 5.2 | For every LCSPlayer in the UserTeam | | | |
| 5.2.1 | Matchup Object | adds | LCSPlayer attributes | Sum |
| 5.3 | Matchup Object | returns | | Sum |
| 5.4 | UserInterface | displays | | Sum for each UserTeam |

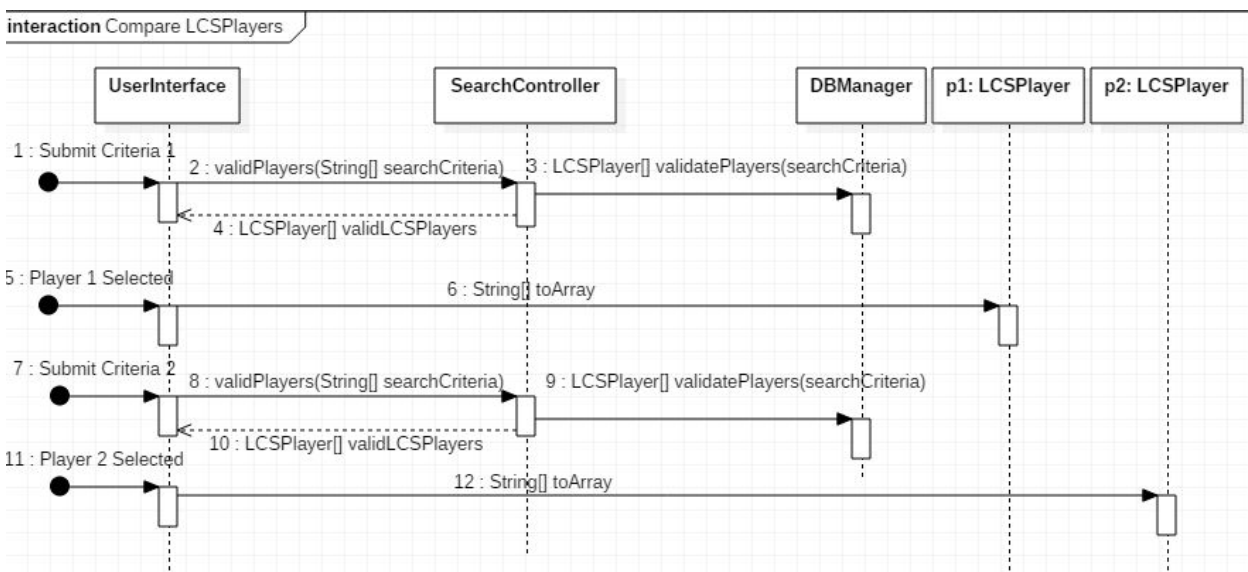
Sequence Diagram

(paste the image of the sequence diagrams under scenario name)

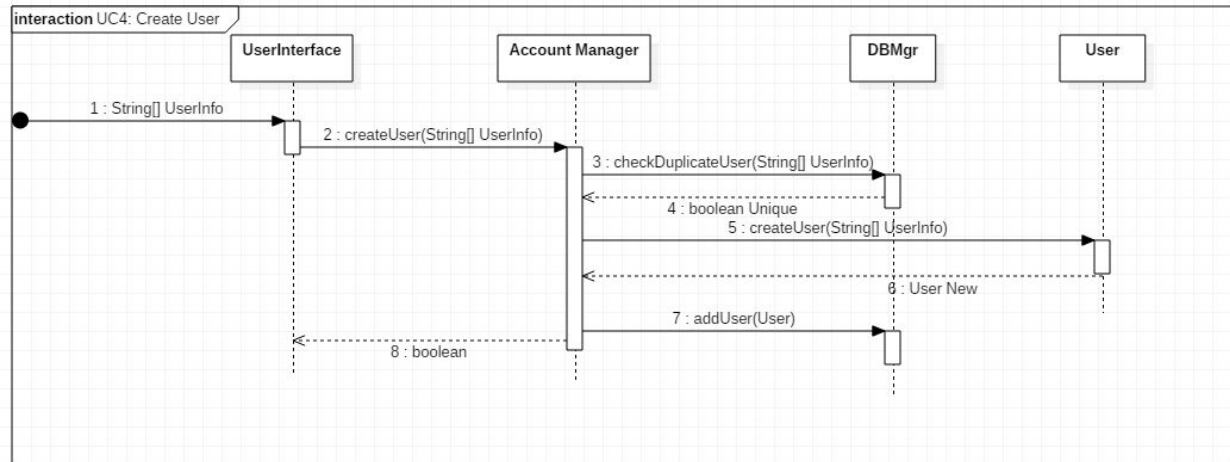
Scenario 1: Database searching (UC 1)



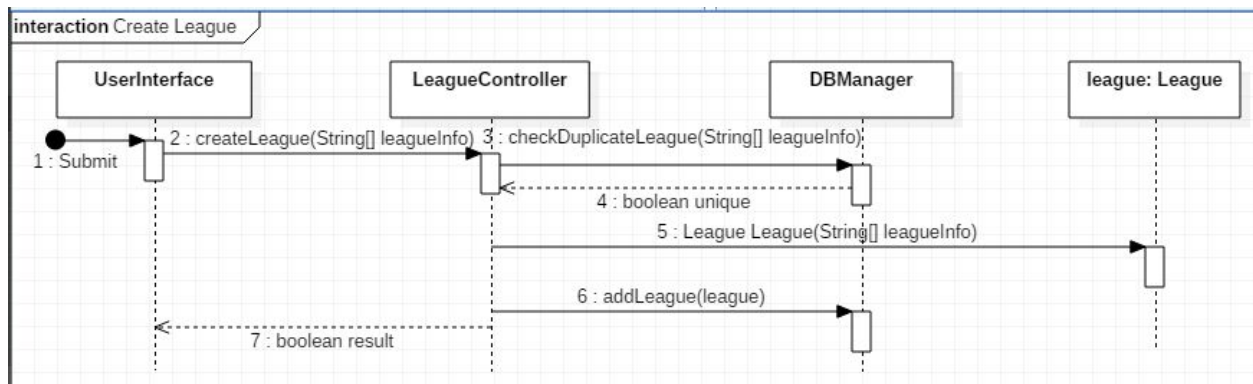
Scenario 2: Compare LCSPlayer (UC2)



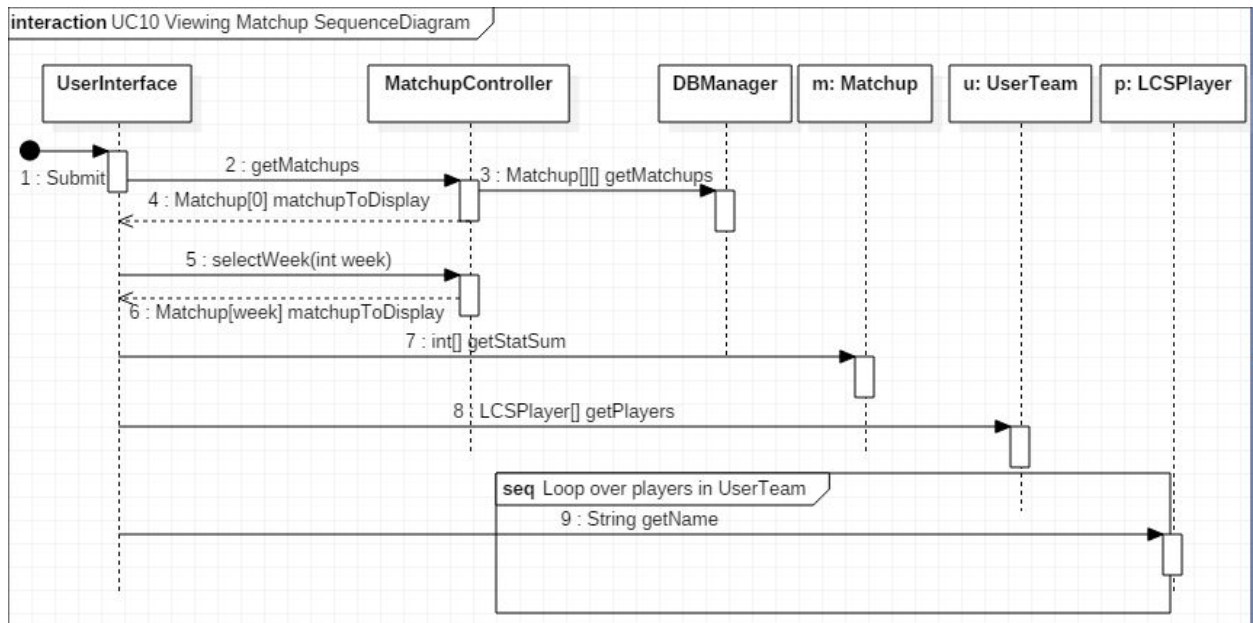
Scenario 4: Create User (UC4)



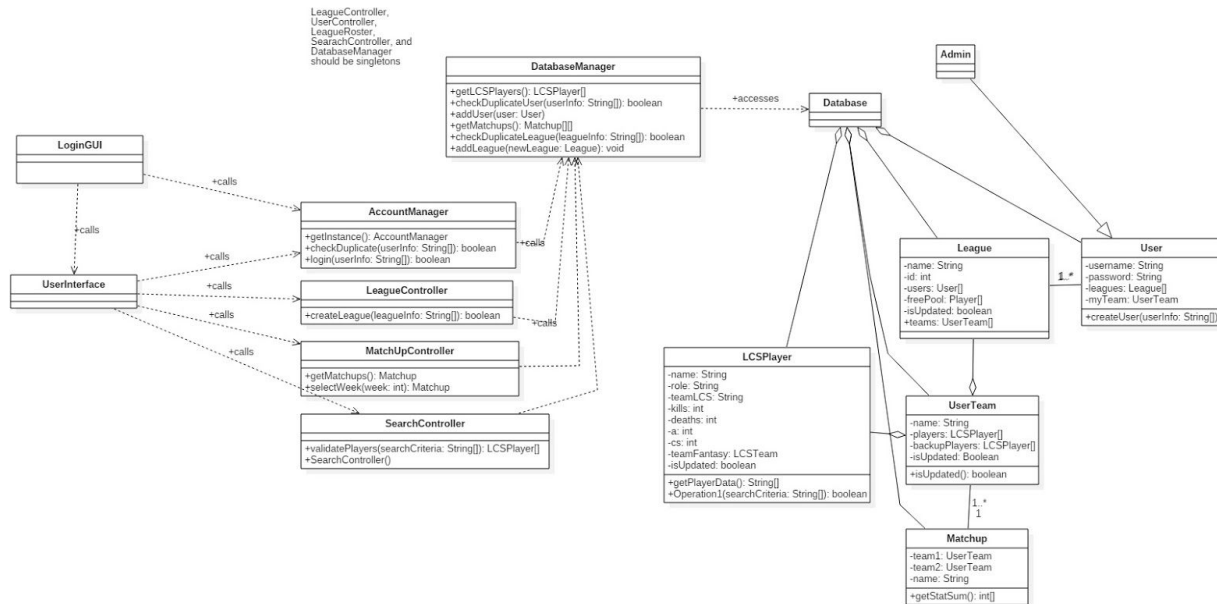
Scenario 5: Create League (UC5)



Scenario 10: View Matchup (UC10)



Design Class Diagram



Hi-res Link:

<https://drive.google.com/file/d/1CLCqb9Sv16e7hhFIO-2tiN76YNOfrkIe/view?usp=sharing>

Database class is intentionally left blank as it is only needed as a reference to link the Database Controller and Database objects (It would be tedious to list all database attributes and functions)

Testing

We are using **black box testing** to test the functionality of our software.

Account Creation

| Test Case | Scenario | Username | Password | Expected Response |
|-----------|-----------------------------|----------|----------|--|
| Case 1 | Successful account creation | Kevin | kevin1 | “Account Created!” message and ability to login. |
| Case 2 | Duplicate account | Kevin | kevin2 | “Account Exists!” message |
| Case 3 | No username or password | | | “Invalid Credentials!” message |

Case 3 originally failed as the checks to validate user input were not programmed in. **This oversight has been rectified.** All other cases passed.

Logging In

| Test Case | Scenario | Username | Password | Expected Response |
|-----------|------------------|----------|-----------|----------------------------------|
| Case 4 | Successful Login | Kevin | kevin1 | Move to main UserInterface |
| Case 5 | Invalid Username | Donkey | kevin1 | “Incorrect Credentials!” message |
| Case 6 | Invalid Password | Kevin | Nosferatu | “Incorrect Credentials!” message |

All test cases passed and are implemented correctly

Using the Matchups tab

| Test Case | Scenario | Selection | Expected Response |
|-----------|--------------------------|------------------|---|
| Case 7 | Selecting a Matchup | Last Matchup | Display the UserTeams of the last Matchup |
| Case 8 | Selecting a Matchup week | Selecting week 8 | Display the Matchups of week 8 |

All test cases passed and are implemented correctly

Using the Search tab

| Test Case | Scenario | Search Input | Expected Response |
|-----------|-----------------------|---|------------------------------|
| Case 9 | Searching a LCSPlayer | Selecting category "Name" Searching "Aphr" | Display LCSPlayer Aphromoo |
| Case 10 | Searching a Role | Selecting category "Role" Searching "Top" | Display Licorice and SSumday |

All test cases passed and are implemented correctly

Using the Compare tab

| Test Case | Scenario | Search Input | Expected Response |
|-----------|----------------------------------|---|---|
| Case 9 | Comparing two LCSPlayers | Selecting category "Name" In left list: Searching "Aphr" Searching "Regin" In right list: Select Aphromoo Select Reginald | Display LCSPlayer Aphromoo and Reginald stats |
| Case 10 | Selecting an LCS not in the list | Invalid input (Not possible) | No response |

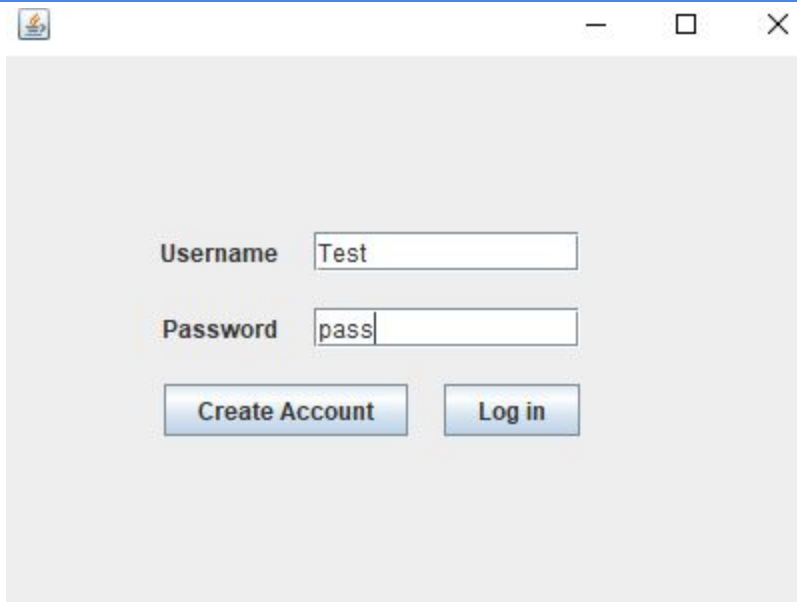
All test cases passed and are implemented correctly

Using the Leagues tab

| Test Case | Scenario | Input | Expected Response |
|-----------|--|---|--|
| Case 11 | Creating a League | Entering in “myTeam” for name Entering in 4 for # of players Selecting “Create League” button | Display new league “myTeam” in list |
| Case 12 | No league name or number of players | Nothing | “Incorrect Input” response |
| Case 13 | Duplicate League names | Entering in “myTeam” for name Entering in 4 for # of players Selecting “Create League” button | “Duplicate League Found” response |

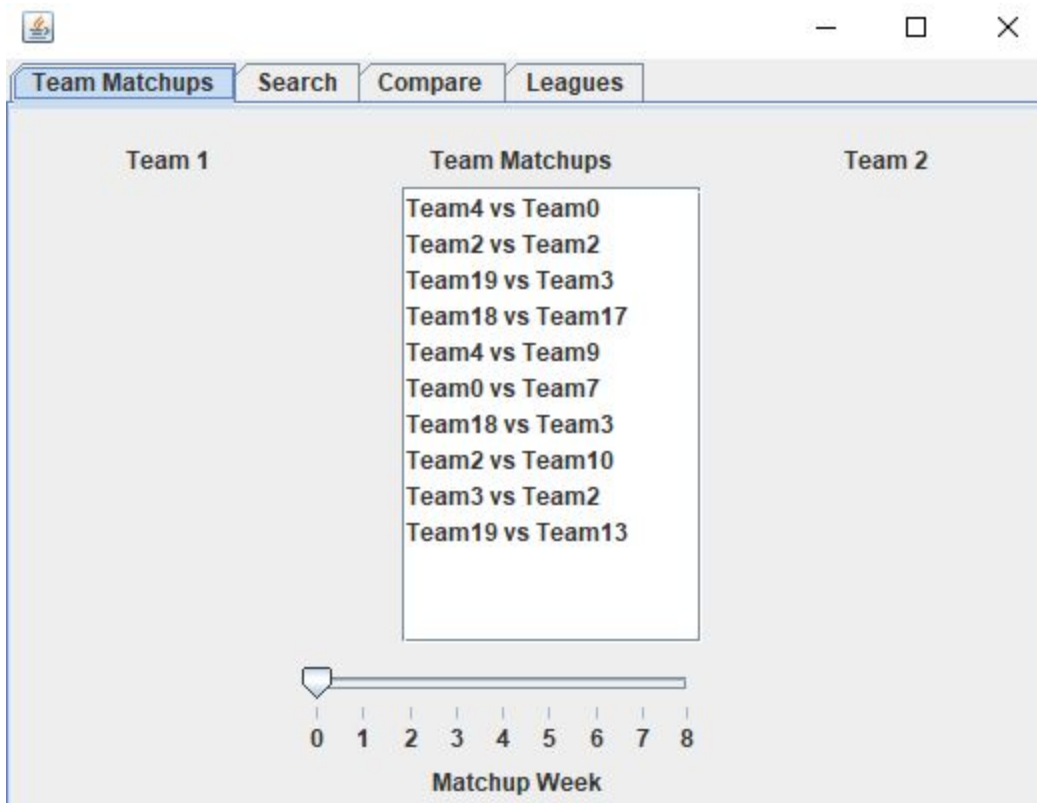
Case 12 failed due to trying to parse an empty string into an integer. **This error has been rectified with input validation.** All other cases passed

Software Demo



A screenshot of a software window with a light gray background. At the top, there is a title bar with a small icon on the left and standard window controls (minimize, maximize, close) on the right. The main area contains a login form. It has two text input fields: the first is labeled "Username" and contains the text "Test"; the second is labeled "Password" and contains the text "pass". Below these fields are two buttons: "Create Account" on the left and "Log in" on the right. Both buttons have a blue gradient and white text.

A username and password will have to be input, and “Create Account” button pressed to create an account. Then the user can log in.



A screenshot of the main user interface of the software. It features a tabbed interface with four tabs: "Team Matchups", "Search", "Compare", and "Leagues". The "Team Matchups" tab is currently selected and active. The main content area is divided into three sections: "Team 1" on the left, "Team Matchups" in the center, and "Team 2" on the right. The "Team Matchups" section contains a list of matchups: Team4 vs Team0, Team2 vs Team2, Team19 vs Team3, Team18 vs Team17, Team4 vs Team9, Team0 vs Team7, Team18 vs Team3, Team2 vs Team10, Team3 vs Team2, and Team19 vs Team13. Below the list is a horizontal slider control with a handle at position 0 and a scale from 0 to 8. The label "Matchup Week" is centered below the slider.

Main UserInterface with Matchup Tab, Search, Tab, Compare Tab, Leagues Tab. Most options in lists will have to be selected before data is displayed. When searching in a text field, the enter bar must be pressed to begin operation.