# Final Project for SW Engineering Class CSC648-848-05 Spring 2024

## Application Title & Name : Teamup

## Team : 5

| Student Name | SFSU Email | GitHub | Discord | Role |
|---|---|---|---|---|
| Juan Estrada | jestradazuluaga@sfsu.edu | jjestrada2 | juanjosee | Team-lead |
| Areeb Abbasi | aabbasi@sfsu.edu | areeeeb | _xertz | Backend-lead |
| Cole Chiodo | cchiodo@sfsu.edu | colechiodo | colechiodo | Docs-editor |
| Jaycee Lorenzo | jlorenzo3@sfsu.edu | jclorenz0 | __jaycee | Frontend-lead |
| Martin Pham | mpham8@sfsu.edu | mar10fam | marnoki | Github-master |
| Kotaro Iwanaga | kiwanaga@sfsu.edu | iamkotaaa | kotaro8448 | Database-admin |

## Milestone 5

Product Link: http://44.220.154.73/

Presentation Link: https://www.youtube.com/watch?v=y1tM-TxPegw

| Version # | Submission Date |
|---|---|
| M5 | 05/23/2024 |
| M4V2 | 05/22/2024 |
| M4V1 | 05/16/2024 |
| M3V2 | 05/15/2024 |
| M3V1 | 04/25/2024 |
| M2V2 | 04/18/2024 |
| M2V1 | 04/04/2024 |
| M1V2 | 03/21/2024 |
| M1V1 | 03/01/2024 |

# Product Summary

- **Product:** TeamUp

- **Major Functions:**

### User:

1. Users shall be able to search for games based on sport type.
2. Users shall be able to search for games based on location.
3. Users shall be able to search for games based on time.
4. Users shall be able to join a team in a game.
5. Users shall be able to create new games.
6. Users shall be able to specify the sport for a game.
7. Users shall be able to specify the location of a game.
8. Users shall be able to specify the time of a game.
9. Users shall be able to specify the number of players needed for a game.
10. Users shall be allowed to update their account profile picture.
11. Users shall be allowed to update their account biography description.
12. Users shall have the option to manually log out from their accounts.
13. Users shall have the option to verify their accounts through email.
14. Users shall be able to recover their password through email.
15. Users shall be able to detach from a game.
16. Users shall be able to search all their joined games.
17. Users shall be able to search all their hosted games.

### Game listings:

1. Game listings shall be able to provide a facility location map.
2. Game listings shall be able to provide the player's username.
3. Game listings shall be able to provide player’s pictures.
4. Game listings shall be able to provide the player’s link to their biography.
5. Game listings shall be able to provide facility game rules.
6. Game listings shall be able to provide the name of the organizer.
7. Game listings shall be able to provide the name of the game.
8. Game listings shall be able to provide the day and time of the game.
9. Game listings shall be able to provide the contact information of the organizer.
10. Game listings shall be able to provide a short description of the game.
11. Game listings shall be able to provide the empty slots of the game.

### Game Location:

1. Game locations shall be able to show address and facility details.
2. Game locations shall be able to provide parking information.
3. Game locations shall be able to provide users with reviews.

### Sport:

1. Users shall be able to choose sports preferences
2. Users shall be able to select their skill level in a specific sport.

### Team:

1. Teams shall be able to be formed in a game listing.
2. Teams shall be restricted by a specific number of players in a game.

### Profile:

1. Profile preferences shall be able to be edited by the user.
2. Profile pictures shall be able to be uploaded to the user's bio(JPG, 320x320px).
3. Profile email shall be able to be edited by the user.
4. Profile phone number shall be able to be edited by the user.
5. Profile gender shall be able to be edited by the user.
6. Profile birthday shall be able to be edited by the user.
7. Profile password shall be able to be edited by the user.

### Review:

1. Users shall be able to rate other players out of 5 stars.
2. Users shall be able to write reviews for other players.
3. Users shall be able to rate facilities out of 5 stars for game locations.
4. Users shall be able to write reviews for game locations.

## • Unique Features:

The unique feature of our app is the ability to use the app to find and join casual sports matches. Our competitors, such as SportsEngine and ZogSports are only for sport tournaments. Our app allows the ability to create and join casual, non-competitive sport matches. Other competitors, like Plei, only focus on a single sport, Soccer. This does not cover casual matches for other sports. And websites like Meetup allow any type of meetup, not just sports. Our app focuses just on sports so it has a better experience for sports.
Another unique feature is the ability to review game locations and users. You can review game locations on aspects such as facility quality or nearby amenities. You can also review other users based on their skill level or attitude. Some of our competitors do offer the ability to give accolades to other users, but not to give a star rating and written review.

## • Product URL: http://44.220.154.73/

# Milestone documents – M1-M4

SW Engineering CSC648-848-05 Spring2024

application title & name : Teamup

Team : 5

| Student Name | Student SFSU Email | GitHub | Discord | Role |
|---|---|---|---|---|
| Juan Estrada | jestradazuluaga@sfsu.edu | jjestrada2 | juanjosee | Team-lead |
| Areeb Abbasi | aabbasi@sfsu.edu | areeeeb | _xertz | Backend-lead |
| Cole Chiodo | cchiodo@sfsu.edu | colechiodo | colechiodo | Docs-editor |
| Jaycee Lorenzo | jlorenzo3@sfsu.edu | jclorenz0 | __jaycee | Frontend-lead |
| Martin Pham | mpham8@sfsu.edu | mar10fam | marnoki | Github-master |
| Kotaro Iwanaga | kiwanaga@sfsu.edu | iamkotaaa | kotaro8448 | Database-admin |

Milestone 1

02/20/2024

| Version # | Date Submitted |
|---|---|
| 2 | 03/21/2024 |
| 1 | 03/01/2024 |
| | |

# Executive Summary

In a world driven by connectivity, our web application, TeamUp, emerges as the ultimate platform for sports enthusiasts at our University. Motivated by the mission of creating a dynamic community of sports lovers, TeamUp is the bridge that connects players, facilitates game exploration, and empowers universities to organize vibrant sports tournaments. This application provides a seamless experience for users to explore, join, and create games effortlessly. With TeamUp's user-friendly interface, individuals can navigate through game schedules, join or form teams, and participate in university-hosted tournaments. For universities, TeamUp streamlines the tournament management process, from creation to participant registration and real-time standings display. This project's value lies in its ability to cultivate a thriving sports community, providing a centralized hub for sports enthusiasts and universities to connect, play, and celebrate the spirit of sportsmanship. It ensures the creation of an innovative, inclusive, and unparalleled sports ecosystem that caters to both the individual player and the university.

# Main Use Cases

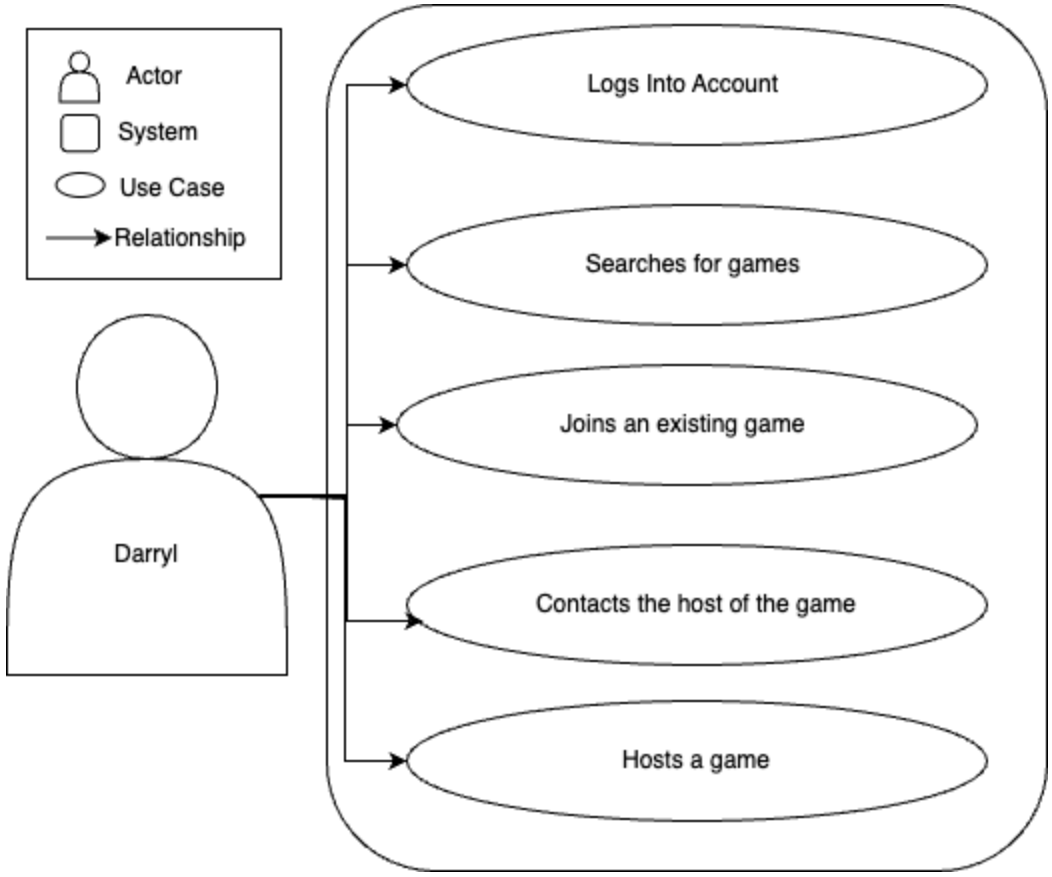
## Use Case 1

- **Actors:** Darryl (1st Year Student), TeamUp (web application)
- **Assumptions:**
    - Darryl is a first year student at SFSU
    - Darryl is interested in sports
- **Use Case:**

Darryl has graduated high school and is heading into his first year of college at SFSU. He's moved here from far away and has not met any people yet, or joined any clubs. He wants to find a way to meet some friends with similar interests as him. While doing some research on SFSU's resources in making connections, he comes across TeamUp. From the convenience of home, he is able to find a group of three people that are hosting a game of soccer to play casually. He decides to attend the game by joining one of the teams on the app. They meet and have a blast playing a small game of soccer. They exchange contacts and keep in touch. Darryl is glad that he was able to easily make some friends using TeamUp.

**Benefits for Darryl:**

- Expands his social circle. His positive experience encourages him to become an active participant on the platform.
- Allows for Darryl to take advantage of being able to initiate contact with potential friends in a low-pressure environment.



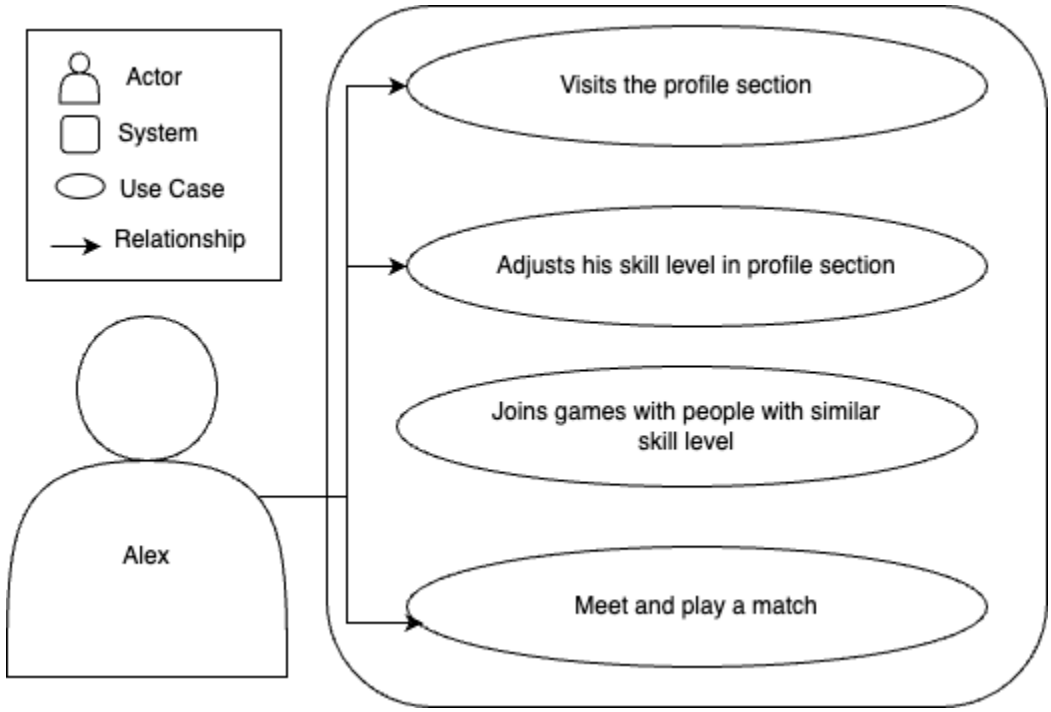

## Use Case 2

- **Actors:** Alex (College Student at SFSU), TeamUp (web application)
- **Assumptions:**
  - Alex is a member already on the app
  - Alex had not selected a skill level when originally creating an account
- **Use Case:**

Alex has played a couple of soccer games at SFSU with some students through TeamUp, but unfortunately he has not found those experiences enjoyable. He finds the gap in the skill levels between him and the people he plays with too big. He believes he is not as skilled as the players that he is playing with, and comes to the conclusion that he needs to find people that are closer to his level. He quickly realizes that TeamUp offers an easy and seamless method of adjusting his skill level. After adjusting his skill level, he finds matches with people of similar skill levels and begins playing games with them. He finds these matches much more enjoyable and competitive, and is thankful for TeamUp's awesome way of showcasing skill levels of players.

**Benefits for Alex:**

- Alex can seamlessly explore and engage in a variety of sports activities, from competitive leagues to casual games, ensuring he finds the perfect fit for his interests and skill level.




## Use Case 3

- **Actors:** John (Gym Supervisor), Mitchell (3rd Year Student), TeamUp (web application)
- **Assumptions:**
  - John and Mitchell are friends.
  - John works at the gym located on campus.
  - John and Mitchell are interested in basketball.
  - John and Micthell attend SFSU.
- **Use Case:**

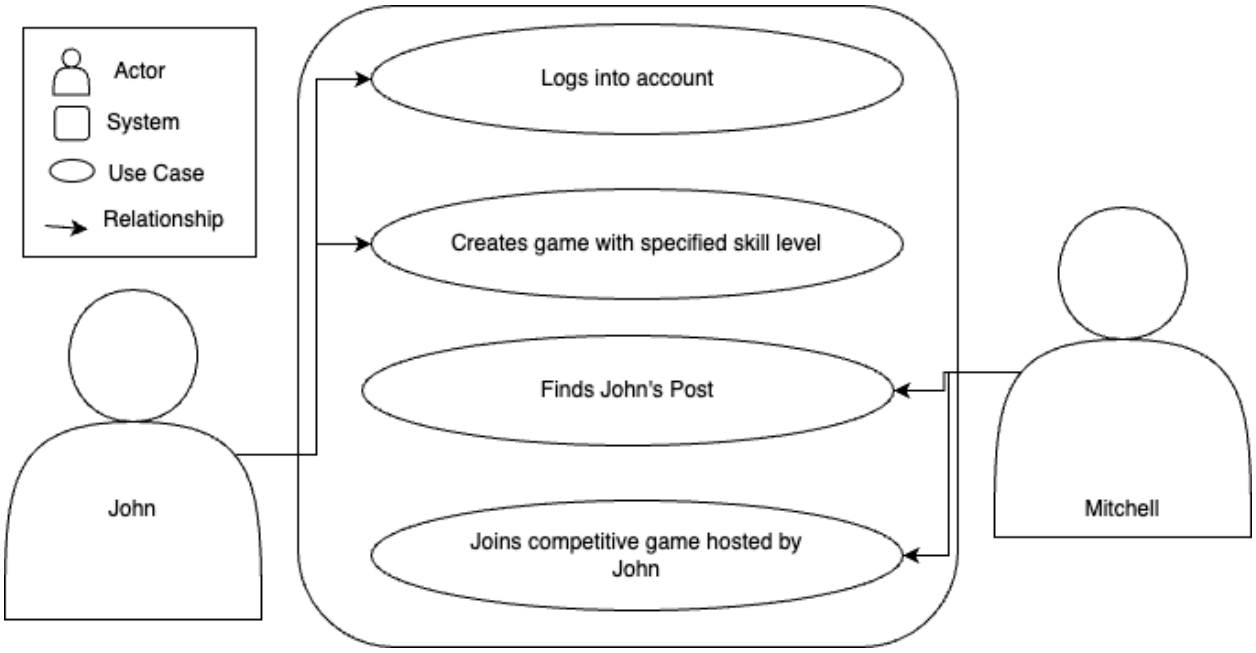
John works at the campus gym at SFSU. Since working there, John has been looking for ways to play basketball with others at a higher skill level. As John is working, he encounters Mitchell as he walks into the gym. John mentions the idea of arranging games at a specific skill level at the gym, but explains the frustrations and difficulties of organizing such events. Mitchell mentions TeamUp as he uses it to find basketball games and explains the features involving hosting games at a specified skill level. After this conversation, John visits TeamUp and is amazed by its ease of use and immediately sets up a basketball game with a skill level of intermediate and higher. Mitchell sees the posting of the game and joins it shortly after. John and Mitchell are glad that they are able to play basketball in a more competitive setting.

**Benefits for John:**

- John can easily coordinate games with specified ~~rules and limits~~ skill levels.

**Benefits for Mitchell:**

- Mitchell can now play basketball more competitively, incentivising him to play at best and become a better player.



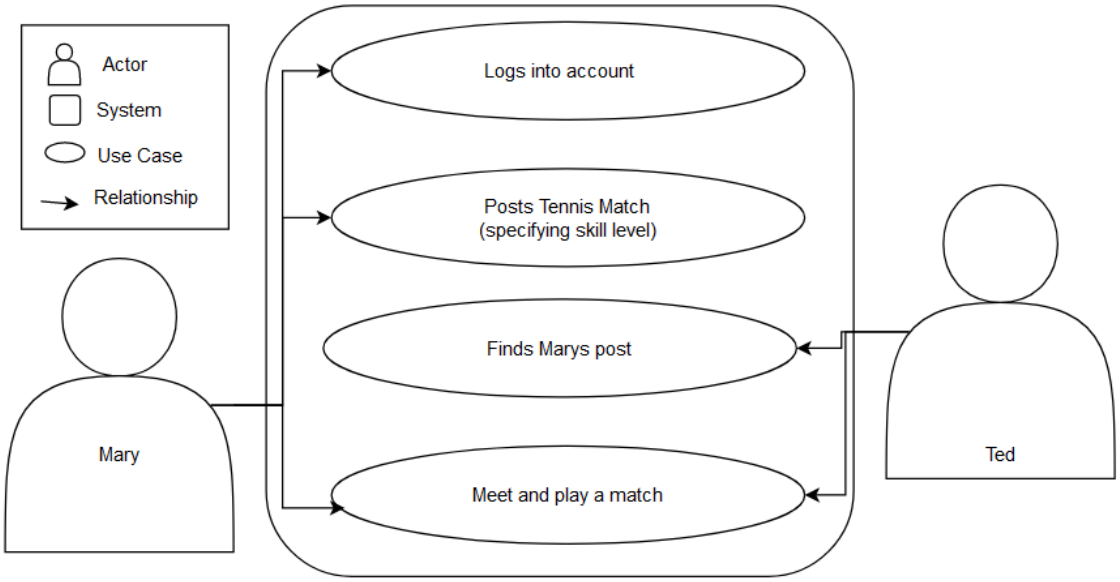

## Use Case 4

- **Actors:** Mary (3rd Year Student), Ted(TeamUp User), TeamUp (Web Application)
- **Assumptions:**
  - Mary is a 3rd year college student
  - Mary is already in the school tennis club
- **Use Case:**

Mary is a 3rd year college student who currently already plays for the school's tennis club. This club is very competitive and attends many tournaments every year. However, Mary would like to have extra practice before her next tournament so she can be as prepared as possible. Because the school tournaments she already attends use an app called TeamUp, she tries to use the other feature on the app. Mary posts on the app that she is looking for tennis opponents who are a certain skill level. Another user of the app, Ted, finds her post and they both meet to play tennis against each other. After the practice session, Mary feels satisfied and has found weaknesses in her game that she can work on.

**Benefits for Mary:**

- Can find opponents to practice against who are a high enough skill level that she desires to practice against.
- She becomes as prepared as she can be before a big tournament.



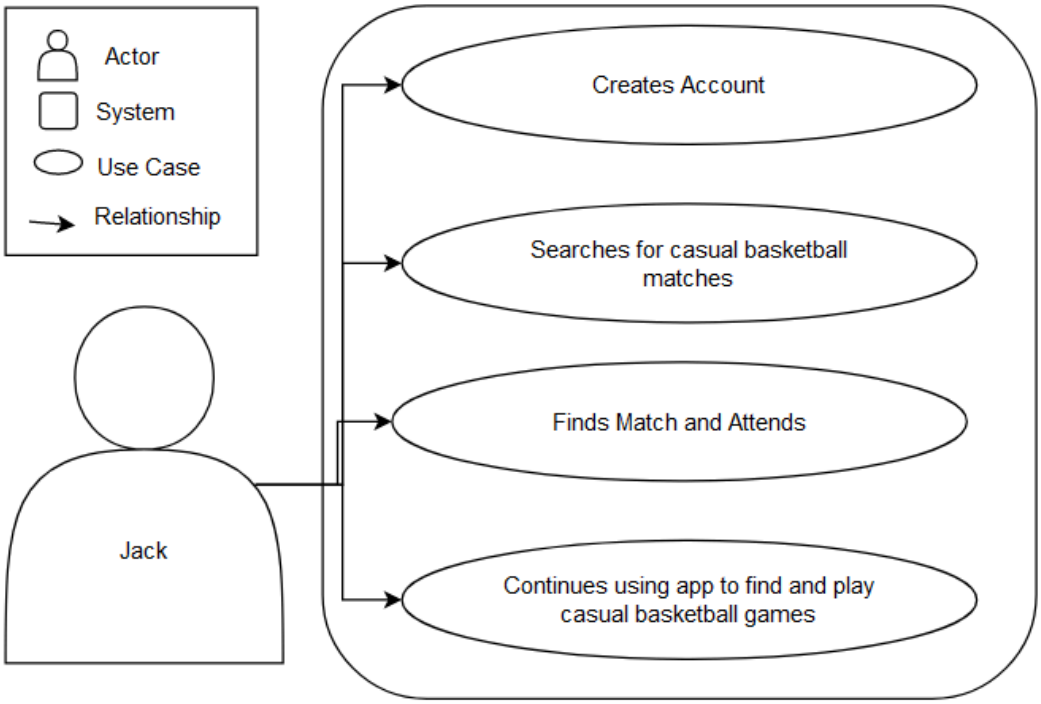

# Use Case 5

- **Actors:** Jack (3rd Year Student), TeamUp (Web Application)
- **Assumptions:**
    - Jack is a 3rd year college student
    - Jack likes to play basketball with his friends sometimes on the weekends
- **Use Case:**

Jack used to play basketball with his friends occasionally on the weekends. However, Jack moved to another state for college, and now does not have friends to play basketball with anymore. Although he likes and plays basketball often, he is not interested in trying out for his new school's basketball team, he wants a more casual experience. Luckily for him, he finds an app called TeamUp. With the app, he finds there are other students who have signed up and have scheduled a casual game in the school's gym. He signs himself up for the game, and now he has found a new group of people to play basketball with whenever he wishes.

**Benefits for Jack:**

- Found a new group of friends that Jack can now play basketball with regularly.




# Use Case 6

- **Actors:** Chris(3rd Year Student), Tyler(2nd Year Student), TeamUp (Web Application)
- **Assumptions:**
    - Chris and Tyler are college roommates.
    - Chris really likes basketball and plays often.
    - Tyler has never played basketball but wants to play.
- **Use Case:**

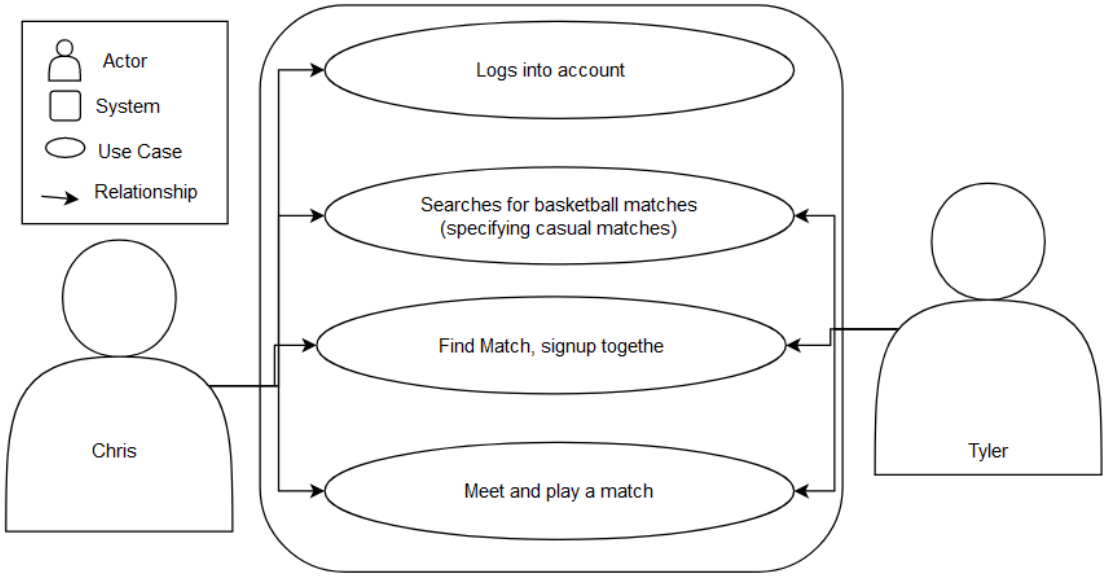
Chris (a 3rd year student) and Tyler (a 2nd year student) are roommates on their college campus. Chris plays basketball nearly every day in the school's gym. Tyler wants to also join, but he has never played and doesn't even know where to start. Chris tells him that he has been using an app called TeamUp to find basketball games to play. Tyler downloads the app, but he is still nervous about playing with strangers. So, Chris offers to join a game with him. On the TeamUp app, they find a casual game being hosted at their school's basketball court. They sign up together and attend the game.

**Benefits for Chris:**

- Chris and Tyler now have a new activity to deepen their friendship.

**Benefits for Tyler:**

- Tyler is more comfortable looking for and playing basketball games on the app.




# Use Case 7

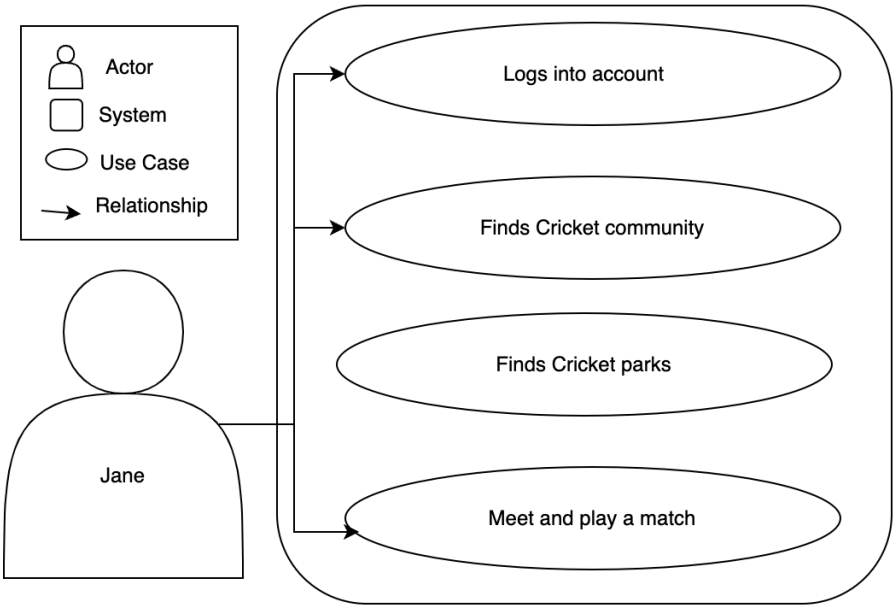
- **Actors:** Jane (San Francisco Resident), TeamUp (Web Application)
- **Assumptions:**
    - Jane has recently moved to San Francisco
    - Jane plays cricket.
- **Use Case:**

In Jane's free time, she likes to play cricket. In the previous city she was living in, there was a large cricket community and games would not be hard to find locally. However, since moving to San Francisco, Jane has not been able to find anywhere to play cricket or any other residents that play the sport as well. After conducting some research online, she stumbles upon a web application called TeamUp. She sees that there is an established cricket community in San Francisco on the app and creates an account. Shortly after creating her account, Jane is able to find facilities and parks where they play her beloved sport. Jane is now immersed in the cricket community in her city and exclusively uses TeamUp to arrange and find games.

**Benefits for Jane:**

- Jane now has a platform that serves her needs of wanting to play local cricket matches.

- Jane can find matches that coincide with her schedule and preferences.




# Use Case 8

- **Actors:** Trevor (San Francisco Resident), Noah (San Francisco Resident), Jimmy (Public Transportation Rider), TeamUp (Web Application)
- **Assumptions:**
  - Trevor and Noah are friends.
  - Trevor, Noah, and Jimmy play badminton.
  - Jimmy is also a San Francisco resident.
- **Use Case:**

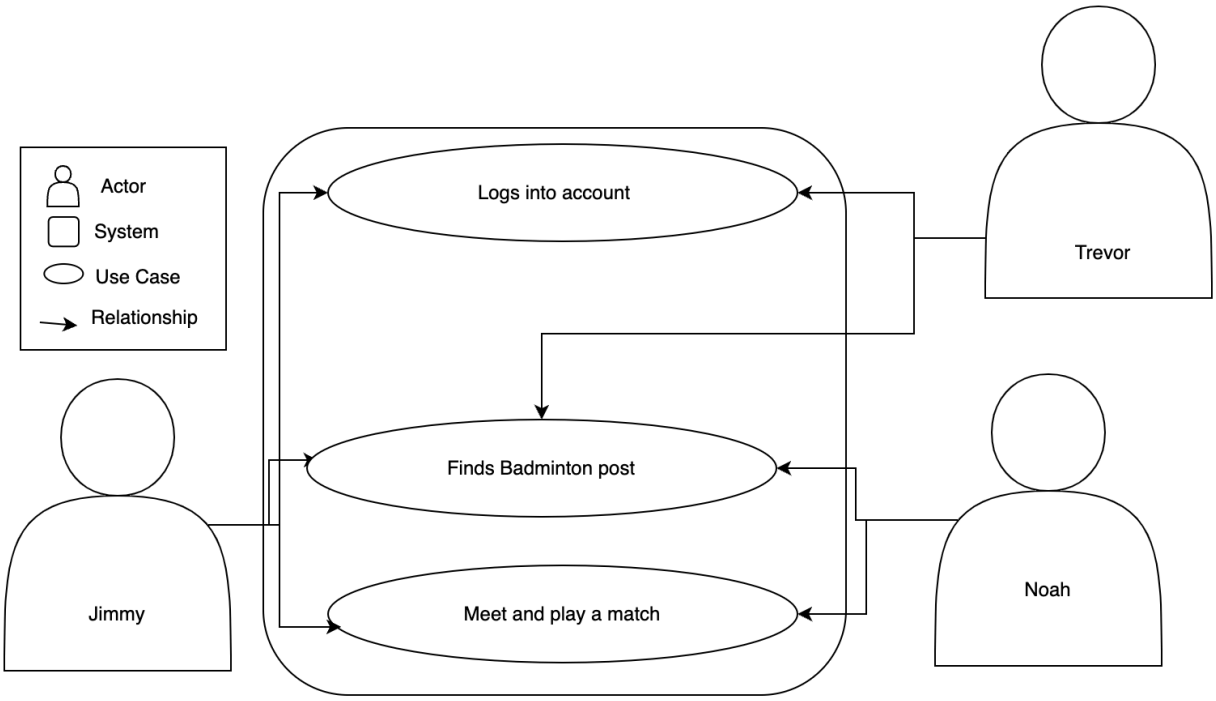
Trevor and Noah are avid badminton players. At times, they want to play doubles but due to the lack of a badminton population in their area, it can be very difficult to find other players that want to play doubles. This problem forces Trevor and Noah to travel across the city to have a chance of finding matches to play against others. Sometimes they commute and face the same issue at their local badminton court, wasting their time. One day when Trevor and Noah are traveling using public transportation, hoping to find a match in another area, they come across another badminton player, Jimmy. After further conversation, Noah expresses their difficulty of finding doubles matches around their local area and having to travel to other courts for a chance of finding other players that want to play. Jimmy introduces Trevor and Noah to TeamUp, which he has been using himself to find games around the city. He explains that TeamUp allows users to find and also set up casual matches for their preferred time and place. Trevor and Noah immediately create accounts and are amazed at the amount of available badminton matches posted spanning the entire city.

**Benefits for Trevor and Noah:**

- Trevor and Noah now can save time and arrange plans to play doubles against other players as opposed to waiting for players to show up to a location or travel to not have a guaranteed chance of playing doubles.

**Benefits for Jimmy:**

- Benefited the badminton community in San Francisco by increasing the population of players on TeamUp.



# Main Data Items and Entities

## 1. User

- Definition: Represents anyone who interacts with the app, including players and school administrators.

## 2. Game

- Definition: A scheduled sports activity that users can join or create.

## 3. Team

- Definition: A group of users who join together to play a sport, either for casual games or within a tournament.

## 4. Tournament

- Definition: A competitive event organized by schools or universities, involving multiple games and teams.

## 5. Match

- Definition: A specific instance of a game within a tournament, between two teams or individuals.

## 6. Location

- Definition: Physical place where games and tournaments are held.

## 7. Profile

- Definition: Detailed information about a user, including their sports preferences and skill levels.

## User Types and Privileges

- Player: Regular users who can browse and join games, create games, and register for tournaments. They can also be part of or form teams.
- School Administrator: Users with the authority to create and manage tournaments on behalf of their institution. They have additional access to tournament management tools.

## Usage in Documentation and Development

- These terms and their definitions will be used consistently across all project documentation, user interfaces, software components, and database designs.
- The division of user types into Players and School Administrators informs the app's functionality and access control mechanisms.
- The attributes listed for each entity provide a high-level overview of the data model and serve as a guide for database design and API development.

## Functional Requirements

**User:**

1. Users shall be able to search for games based on sport type. 1
2. Users shall be able to search for games based on location. 1
3. Users shall be able to search for games based on time.1
4. Users shall be able to search for games based on player skill level. 1
5. User shall join existing games. 1
6. Users shall create new games. 1
7. Users shall specify sport, location, time, and number of players needed for a game. 1
8. Users shall invite friends to join created games. 3
9. Users shall be able to join a game as a team 2
10. Users shall be able to join existing teams. 2
11. Users shall be able to join registered tournaments 2

**Game:**

1. Game listing shall provide facility location. 1
2. Game listing shall provide player profiles. 1
3. Game listing shall provide facility game rules. 1

Tournament:

1. Users shall receive confirmation emails containing tournament details.
2. Users shall receive notifications containing tournament participation instructions.
3. Users shall receive reminders to registered participants before the tournament start date.
4. Team members shall be able to communicate to coordinate logistical issues related to games.
5. Admins shall have the ability to update or change the tournament schedules.
6. Tournaments shall generate games schedules based on the number of teams, available time slots, and tournament format.
7. Participants shall receive notifications of their match schedules, including date, time, opponent, and location.
8. The tournament shall enforce a code of conduct policy for game participants.
9. Tournament violations of the code of conduct shall be reported.
10. The game results including scores and winners, shall be updated within the app.
11. The app shall recognize the outstanding performances of top participants.
12. Users shall have the opportunity to provide feedback and evaluations on the game.

User Authentication and Authorization:

1. The system shall enforce password strength requirements.
2. Passwords shall be securely hashed and stored in the database.
3. The system shall support optional two-factor authentication (2FA).
4. Users shall have the ability to recover forgotten passwords.
5. The system shall allow users to update their account information.
6. Users shall have the option to manually log out from their accounts.

# Non-Functional Requirements

## Reliability:

- The system shall be available 99.9% of the time during peak usage hours.
- The application shall be resilient to server failures and able to recover gracefully without data loss.

## Response Time:

- The system shall respond to user interactions within 2 seconds for browsing games and tournaments.
- User actions such as joining a game or registering for a tournament shall complete within 5 seconds.

## Hardware and Networking Requirements:

- The application shall be hosted on servers with sufficient processing power and memory to handle concurrent user requests.
- Network bandwidth shall be adequate to support simultaneous interactions from multiple users.

## Usability Requirements:

- The user interface shall be intuitive and easy to navigate for users with basic computer literacy.
- User interactions and workflows shall be consistent across different sections of the application.

## Privacy:

- User data collected shall include name, email address, and sports preferences.
- Data collected shall only be used for facilitating game and tournament management and shall not be shared with third parties without user consent.

## Compatibility:

- The application shall be compatible with major web browsers such as Chrome, Firefox, Safari, and Edge.
- Responsive design principles shall be implemented to ensure optimal viewing and usability across desktop and mobile devices.

## Media Content:

- Images and media files uploaded by users shall adhere to specified formats and size limits to ensure efficient storage and retrieval.
- Accepted file formats and size limits shall be communicated to users during the upload process.

## Performance:

- The application shall be capable of handling concurrent user interactions without significant degradation in performance.
- Database queries and server-side operations shall be optimized to minimize latency and ensure smooth user experience.

## Licensing and Legal:

- The application shall comply with relevant laws and regulations governing data privacy, user consent, and intellectual property rights.
- Proper licensing for third-party libraries and software components used in the development shall be obtained and documented.

## Training and Support:

- Users with a high-school diploma, after 1 hour of training, shall be able to navigate and use core features of the application.
- Support documentation and resources shall be provided to assist users in troubleshooting common issues and accessing help when needed.

## Tools and Requirement Management:

- Requirements management shall be handled using tools from XYZ, ensuring traceability and accountability throughout the development lifecycle.
- Each requirement shall be associated with identifiable portions of code referenced by module name and code line number for ease of maintenance and debugging.

# Competitive analysis

1. SportsEngine Tourney
    - SportsEngine Tourney is a website/application for easily managing sports tournaments, competitions, and leagues. It has about 380,000 reviews on the app store. More than one million games are played annually.

2. Piel | Pick Up Soccer
    - Plie | Pick Up Soccer is an application for people who want to get together and play soccer casually.Users can find games that are held nearby. There are 5300 ratings on the app store.

3. Meetup
    - Meetup is an online platform where people can join communities with common interests and hobbies. It's the most famous and has the most users among these five competitors. Meetup has more than 60 million registered users and there are 330,000 groups in 193 countries and 10,000 cities.

4. MaxPreps
    - MaxPreps is an online sports media focused on high school sports. They provide game results, game schedules, and statistics for high school sports. It covers a wide range of sports such as baseball, basketball, golf, and soccer, etc.

5. ZogSports
    - ZogSports is a website/application that offers leagues in various sports for adults, mainly in major cities. There are leagues for major sports such as basketball and soccer as well as minor sports such as pickleball and kickball.

6. Reclub
    - You can participate in sports communities and competitions on Reclub. You can also find coaches as well as sports matching. It's used in many English speaking countries and has over 10k downloads on Google Play.

| Company | Reclub | Meetup | Plei | SE tourney | MaxPreps | ZogSports |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| Strength | Many active users and groups<br><br>Users can find coaches<br><br>Many sports are available<br><br>Great UI | Many users/communities<br><br>Great Interface, good UI<br><br>It is easy to use the website/application | Focusing on soccer.<br><br>Many followers on social media such as Instagram and FaceBook<br><br>Easy to create an account<br><br>Users can invite friends by using your phone number<br><br>It automatically creates a team for you | easy to manage competitions, tournament, and leagues | Focusing on high school athletes.<br><br>users can view news, rankings and statistics.<br><br>Users can use the services without creating an account | Many users and many active groups |
| Weakness | The sports are defined by the developers<br><br>There is not many casual games<br><br>Games that are full show up in search<br><br>The app displays clubs and meets that have not been active for a certain period of time | Not focused solely on sports, there is no tournament feature | Focusing only on soccer.<br><br>The UI is not the best, the green background is hard to see.<br><br>You need a picture to create an account<br><br>Users have to purchase game credits instead of paying Directly<br><br>There are | Only focused on tournaments and leagues, not many uses of social media | Sports matching is not available.<br><br>Too many ads.<br><br>Focusing on high school athletes limits the number of users. | The main target is adults. Only focusing on leagues. Only available in big cities such as NY, SF, etc. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | not many users. | | | |
| Pricing | Free, but it costs money to participate in games. | Depends on the size of the groups. As the size of groups gets bigger, the price increases | Free, but it costs a few dollars to participate in games | Pricing was unavailable . You have to contact the company | Free | Free, but it costs a few dollars to participate in games. |
| Social media | Instagram, X, FaceBook | FaceBook, X, Instagram | Newsletter, blog, Instagram, FaceBook | Blog | Tiktok, YouTube, Instagram, FaceBook, X | Blog, Instagram |
| Onboarding experience | Smooth instructions | Smooth instructions | Not much support after first step | Not much support from the beginning | Smooth instructions | Smooth instructions |

| Features | Reclub | Meetup | Plei | SE tourney | MaxPreps | ZogSports | Our product |
|---|---|---|---|---|---|---|---|
| Text Search | + | ++ | + | - | + | + | + |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Boolean Search | ++ | ++ | - | + | + | ++ | + |
| Leagues/Tournament | + | - | - | ++ | - | ++ | ++ |
| Availability in School/College | - | - | - | - | + | + | ++ |
| Browse | + | ++ | + | + | - | + | + |

**Summary:**

What we found in our research of the five competitors is that there are not many matching services that allow people to play sports casually. For example, SportsEngine Tourney and ZogSports focus only on leagues and tournaments. There are services for those people, such as Plei and Meetup, but since Plei is only for soccer and Meetup is not only for sports, each has its own disadvantages. Therefore, our product allows users to find people who want to play sports casually, which differentiates us from other similar services. We also found that there are not many school-specific services. Our product allows schools and universities to hold tournaments, which is a unique feature of our product not found in other services. The benefits to schools of using this service are many. For example, it doesn't only help students make friends but also promotes health and helps students release stress. We are looking to raise money by marketing our product to schools. A sports and health version of Canvas is our goal. When we looked at the market, we noticed that there are not many services that utilize social media. Since most students use social media, we are considering marketing through the active use of TikTok and Instagram. People who want to enjoy sports casually and students are our main target audiences.

## Checklist

• Team found a time slot to meet outside of the class. DONE
• Github master chosen. DONE.
• The team decided and agreed together on using the listed SW tools and

deployment server. DONE.
• Team ready and able to use the chosen back and front-end frameworks and those who need to learn are working on learning and practicing. ON TRACK
• The Team lead ensured that all team members read the final M1 and agree/understood it before submission.DONE
• Github organized as discussed in class (e.g. master branch, development branch, a folder for milestone documents etc.)DONE;

## Contributions

| | | |
|---|---|---|
| Juan Estrada | 9 | Submit Milestone One & create the workflow for Milestone 2<br>List of non-functional requirements<br>List of functional requirements |
| Kotaro Iwanaga | 9 | Version 2 of Competitive analysis implementing teammate's opinion |
| Cole Chiodo | 8 | Format the milestone 1 pdf document and make sure we don't miss any sub-checkpoint<br>Main Use Cases |
| Martin Pham | 8 | Merge the "about" branch to main and watch<br>Executive Summary<br>Main Use Cases |
| Jaycee Lorenzo | 9 | Main Use Cases<br>Home page with team members page /template for individual team member single page |
| Areeb abbasi | 8 | Deploy app and share : Website URL<br>Main Data Items and Entities |

# High-level system architecture and technologies used

## Frontend Components:

- JavaScript ECMAScript 2023 (ES14)
- React 18.2.0

## Backend Components:

- JavaScript ECMAScript 2023 (ES14)
- Node.js v21.0.0  with Express.js 4
- MySQL 8.0.36

## Testing Tools:

- Jest 29.7.0

## SSL:

- Cloudflare offers free SSL certificates for websites using their content delivery network (CDN) services.

## Deployment and Hosting Platform:

- Amazon Web Services (AWS)
- Ubuntu Server 23.10

| | React.js(JS) | Node/Express(JS) | AWS | MySQL | Jest |
|---|---|---|---|---|---|
| Juan Estrada | 3 | 3 | 3 | 4 | 3 |
| Cole Chiodo | 2 | 3 | 2 | 4 | 2 |
| Kotaro Iwanaga | 3 | 3 | 3 | 4 | 2 |
| Martin Pham | 4 | 3 | 2 | 4 | 2 |
| Jaycee Lorenzo | 4 | 3 | 2 | 4 | 2 |
| Areeb Abbasi | 3 | 5 | 5 | 5 | 3 |

| Item | Credentials |
|---|---|
| Website URL | http://54.163.16.75/ |
| SSH URL | ec2-54-163-16-75.compute-1.amazonaws.com |
| SSH Username | ubuntu |
| SSH Password | I uploaded the .pem file in the credentials folder |
| Key | I uploaded the .pem file in the credentials folder |
| Database URL | database-648.czcm6osyi4ii.us-east-1.rds.amazonaws.com |
| Database Username | admin |
| Database Password | Jose648# |

SW Engineering CSC648-848-05 Spring 2024

Application Title & Name : Teamup

Team : 5

| Student Name | SFSU Email | GitHub | Discord | Role |
|---|---|---|---|---|
| Juan Estrada | jestradazuluaga@sfsu.edu | jjestrada2 | juanjosee | Team-lead |
| Areeb Abbasi | aabbasi@sfsu.edu | areeeeb | _xertz | Backend-lead |
| Cole Chiodo | cchiodo@sfsu.edu | colechiodo | colechiodo | Docs-editor |
| Jaycee Lorenzo | jlorenzo3@sfsu.edu | jclorenz0 | __jaycee | Frontend-lead |
| Martin Pham | mpham8@sfsu.edu | mar10fam | marnoki | Github-master |
| Kotaro Iwanaga | kiwanaga@sfsu.edu | iamkotaaa | kotaro8448 | Database-admin |

Milestone 2

3/6/2024

| Version # | Submission Date |
|---|---|
| M2V2 | 04/18/2024 |
| M2V1 | 04/04/2024 |
| M1V2 | 03/21/2024 |
| M1V1 | 03/01/2024 |

# 1. Data Definitions

## 1. User

a. Definition: Represents anyone who interacts with the app, including players and school administrators
b. Usage: Users can browse games, join or create games, and participate in tournaments. School administrators have additional privileges to create and manage tournaments.
c. Attributes: User ID, Name, Email, Password, Profile Picture (JPG, 320x320px), Skill Level, Sports Preferences, Availability.

## 2. Game

a. Definition: A scheduled sports activity that users can join or create.
b. Usage: Allows users to find, join, or post games matching their interests and schedules.
c. Attributes: Game ID, Sport Type, Location, Date & Time, Required Number of Players, Skill Level Preference, Equipment Details, Organizer (User ID).

## 3. Team

a. Definition: A group of users who join together to play a sport, either for casual games or within a tournament.
b. Usage: Facilitates team formation for both casual play and competitive tournaments.
c. Attributes: Team ID, Team Name, Members (List of User IDs), Sport Type, Skill Level.

## 4. Tournament

a. Definition: A competitive event organized by schools or universities, involving multiple games and teams.
b. Usage: Schools can create tournaments, and students can register as solo players, groups, or teams.
c. Attributes: Tournament ID, Name, Sport Type, Date & Time, Location, Registration Details, Tournament Format, Brackets, Standings.

## 5. Game Location

a. Definition: Physical place where games and tournaments are held.
b. Usage: Users can select locations for their games or view where tournament matches are taking place.

c. Attributes: Location ID, Name, Address, Facility Details, Parking Information.

## 6. Profile

a. Definition: Detailed information about a user, including their sports preferences and skill levels.
b. Usage: Helps in matching users with appropriate games and teams.
c. Attributes: User ID, Skill Levels (per sport), Sports Preferences, Biography, Contact Information.

## 7. Admin

a. Definition: A user who is associated with one school who can create tournaments.
b. Usage: Admins are approved users with higher privileges. They are able to create tournaments for the school they are associated with.
c. Attributes: Admin ID, User ID, School ID

## 8. School

a. Definition: Location that can host Tournaments.
b. Usage: Locations that Admins can host Tournaments at.
c. Attributes: School ID, Name, Location.

## 9. Sport

a. Definition: The activity that will be played during a game.
b. Usage: When creating a game, a sport is chosen that will be played. Users can also define their sport preferences.
c. Attributes: Sport ID, Name, Description.

## 10. Review

a. Definition: A rating given from one user to another.
b. Usage: Users can rate other users based on their performance or actions after they have played a game together.
c. Attributes: Review ID, Rating, Description, User ID.

## 11. Sport Level

a. Definition: A level of expertise
b. Usage: Users can filter games based on the level of expertise of a specific game.
c. Attributes: User ID, Sport ID, Level.

## User Types and Privileges

● Player: Regular users who can browse and join games, create games, and register for tournaments. They can also be part of or form teams.
● School Administrator: Users with the authority to create and manage tournaments on behalf of their institution. They have additional access to tournament management tools.

## Registration Info

● Upon account registration, the user must provide the following items of information:
    ● Full Name, Username, Email, Password, Date of Birth, Gender, Phone Number.

## Usage in Documentation and Development

● These terms and their definitions will be used consistently across all project documentation, user interfaces, software components, and database designs.
● The division of user types into Players and School Administrators informs the app's functionality and access control mechanisms.
● The attributes listed for each entity provide a high-level overview of the data model and serve as a guide for database design and API development.

# 2. Prioritized Functional Requirements

## Priority 1:

**User**

1. Users shall be able to search for games based on sport type.
2. Users shall be able to search for games based on location
3. Users shall be able to search for games based on time.
4. Users shall be able to search for games based on player skill level.
5. User shall join existing games.
6. Users shall create new games.
7. Users shall specify the sport, location, time, and number of players needed for a game.
8. Users shall be enforced password strength requirements.
9. Users shall be able to use optional two-factor authentication (2FA).
10. Users shall have the ability to recover forgotten passwords.
11. Users shall be allowed to update their account information.
12. Users shall have the option to manually log out from their accounts.
13. User shall be able to customize their profiles with avatars and bios.
14. Users shall be able to use filters by gender for search games.
15. Users shall be able to rate other players based on skill level.
16. Users shall be able to rate facilities for game locations
17. Users shall earn badges for achievements

**Game**

18. Game listing shall provide facility location.
19. Game listing shall provide player profiles.
20. Game listing shall provide facility game rules.

## Priority 2 :

**User:**

21. Users shall be able to join a game as a team
22. Users shall be able to join existing teams.
23. Users shall be able to join registered tournaments
24. Users shall be able to join registered tournaments
25. Users shall be able to join a game as a team
26. Users shall receive confirmation emails containing tournament details.
27. Users shall receive notifications containing tournament participation instructions.
28. Users shall receive reminders to registered participants before the tournament start date.
29. Team members shall be able to communicate to coordinate logistical issues related to games.
30. Admins shall have the ability to update or change the tournament schedules.
31. Participants shall receive notifications of their match schedules, including date, time, opponent, and location.
32. The app shall recognize the outstanding performances of top participants.
33. Users shall have the opportunity to provide feedback and evaluations on the game.

**Tournament:**

34. Tournaments shall generate game schedules based on the number of teams, available time slots, and tournament format.
35. The tournament shall enforce a code of conduct policy for game participants.
36. Tournament violations of the code of conduct shall be reported.
37. Admins shall be able to manage team rosters, set up team events, and send out team announcements.

## Priority 3 :

**User:**

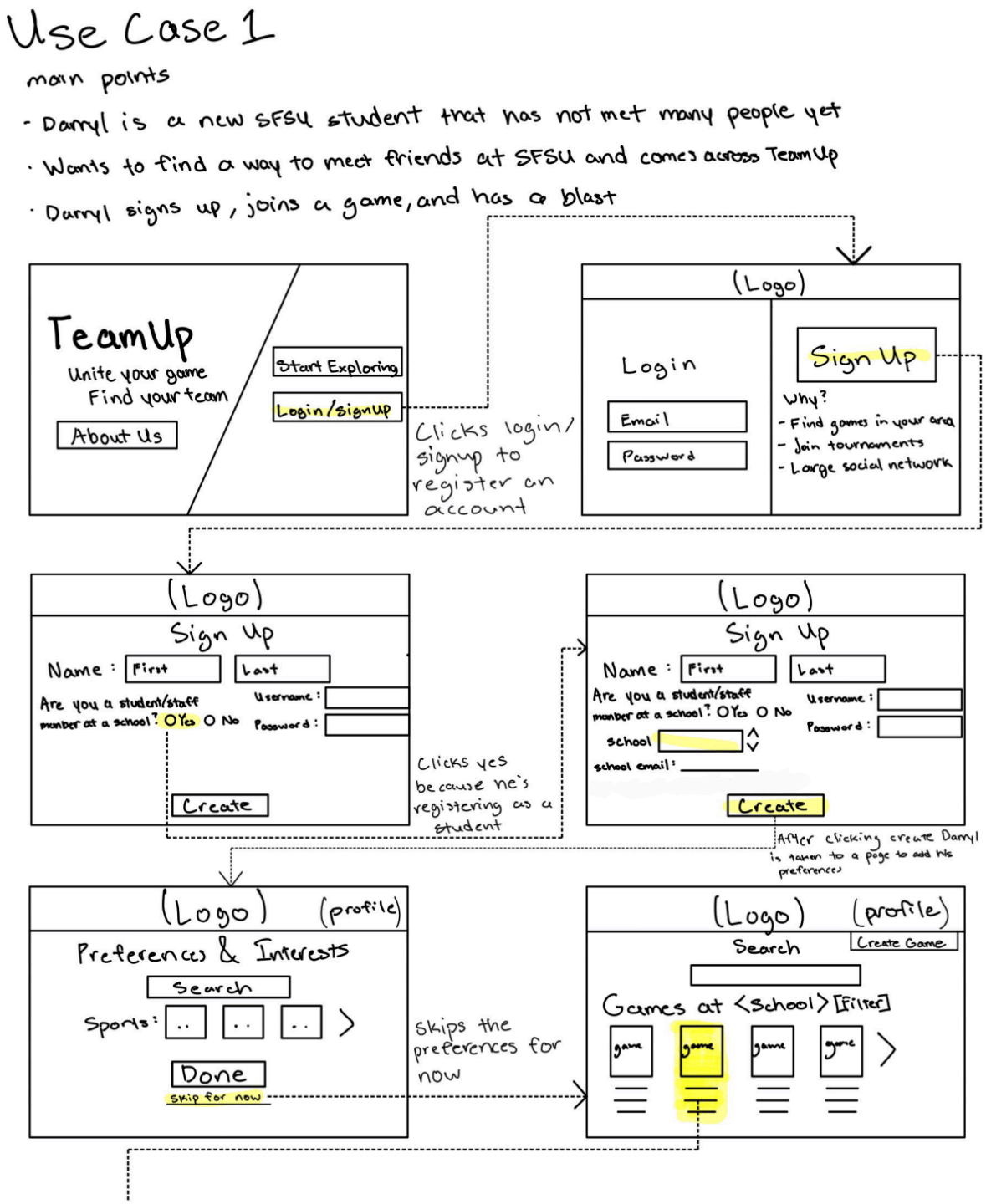
38. Users shall have privacy settings to control who can see their profile and game activity.
39. Users shall receive real-time updates about game changes (like location changes, time adjustments).
40. Users shall see leaderboards for various metrics like most games played,
41. Games shall be able to provide suggestions for indoor locations in case of predicted bad weather.
42. Game listings shall include options for renting necessary sports equipment
43. Game listings shall support calendar integrations for managing team schedules.

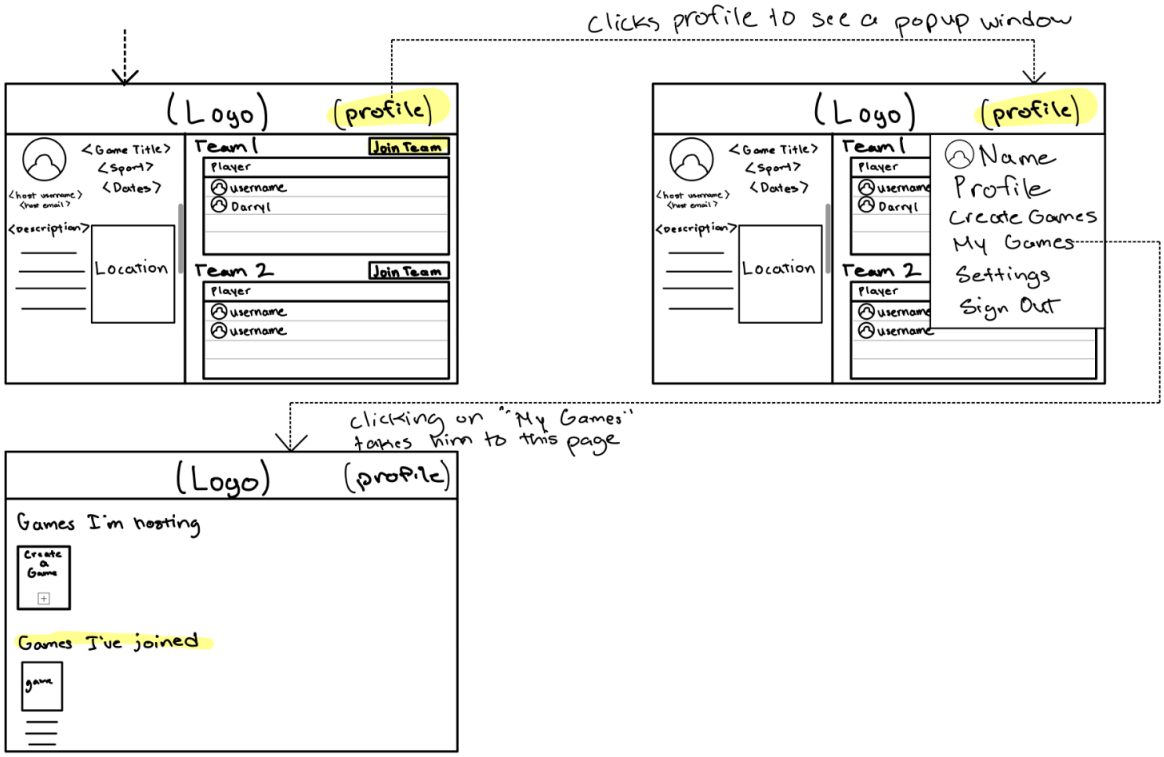
44. Game listings shall include weather forecasts for the scheduled time and location.

**Tournament:**

45. Tournaments shall offer options to livestream games.

# 3. UI Mockups and Storyboards

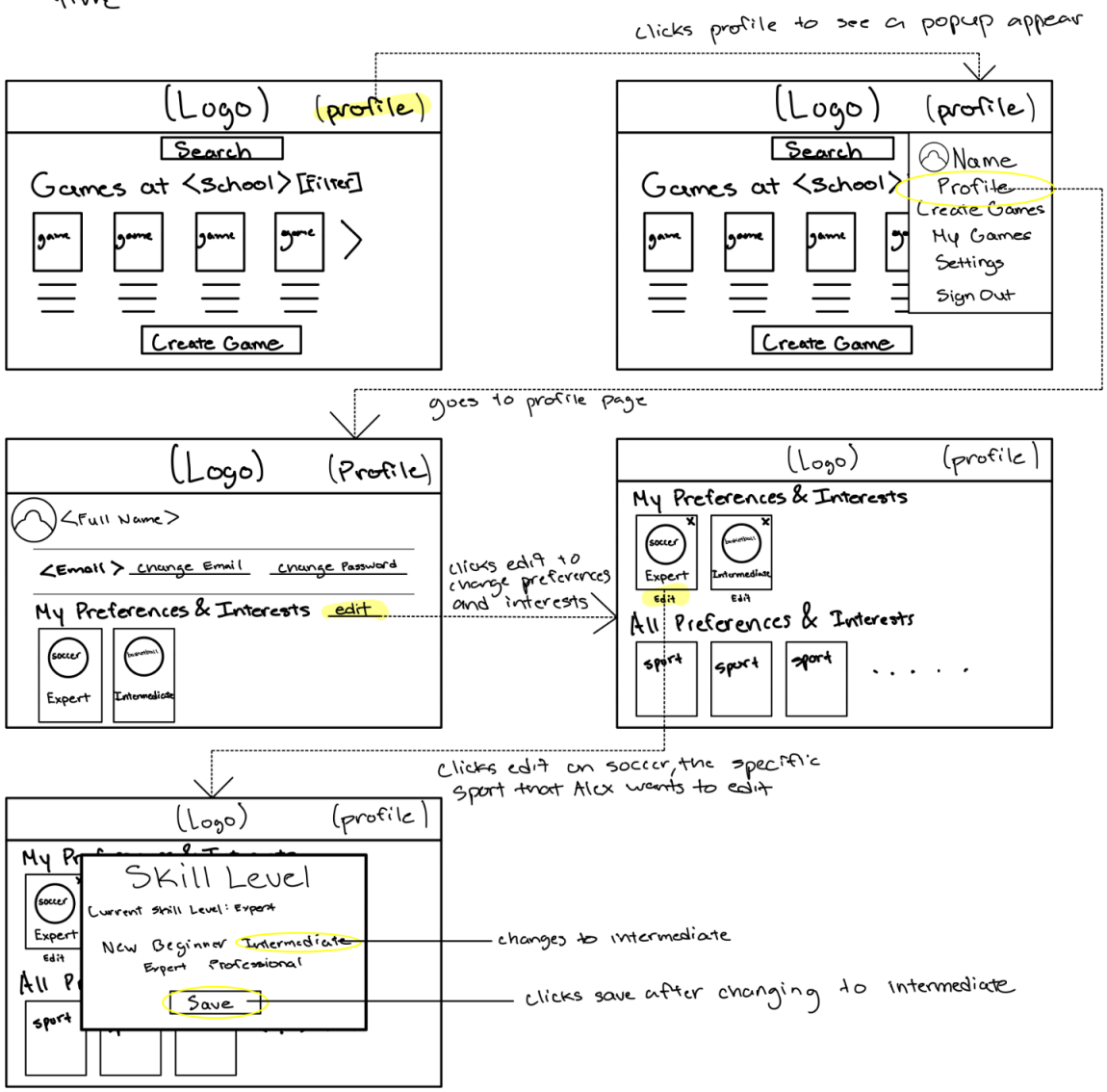



clicks profile to see a popup window
(Logo)
(profile)
<Game Title>
<Sport>
<Dates>
<host username>
<host email>
<description>
Location
Team 1
Join Team
Player
username
Darryl
Team 2
Join Team
Player
username
username
Name
Profile
Create Games
My Games
Settings
Sign Out
clicking on "My Games" takes him to this page
(Logo)
(profile)
Games I'm hosting
Create a Game
Games I've joined
game

# Use Case 2

main points:

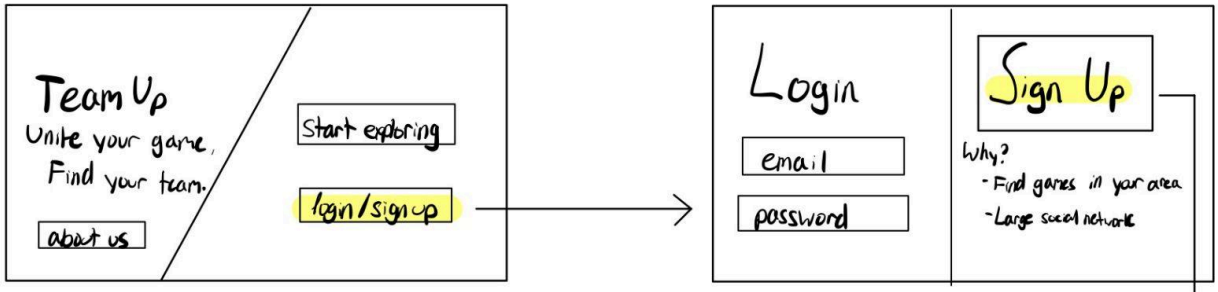
- Alex is already a member on TeamUp
- After playing some games, he believes its necessary to adjust his skill level
- Once he adjusts his skill level, he has a much more enjoyable time



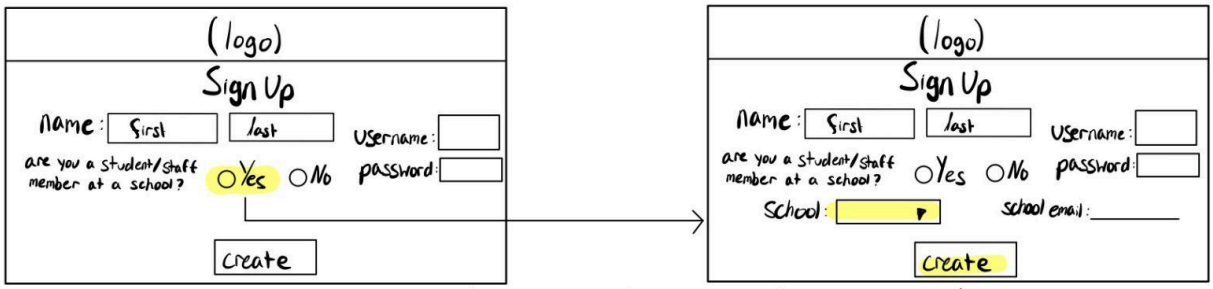
# Use Case 3

main points:

- John wants to set up basketball games with other high skill level players.
- John gets introduced to TeamUp.
- John registers and creates a game catered to more experienced players.

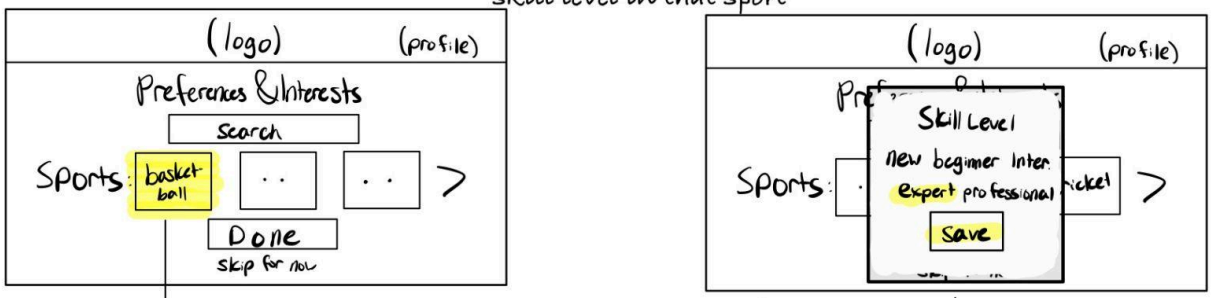


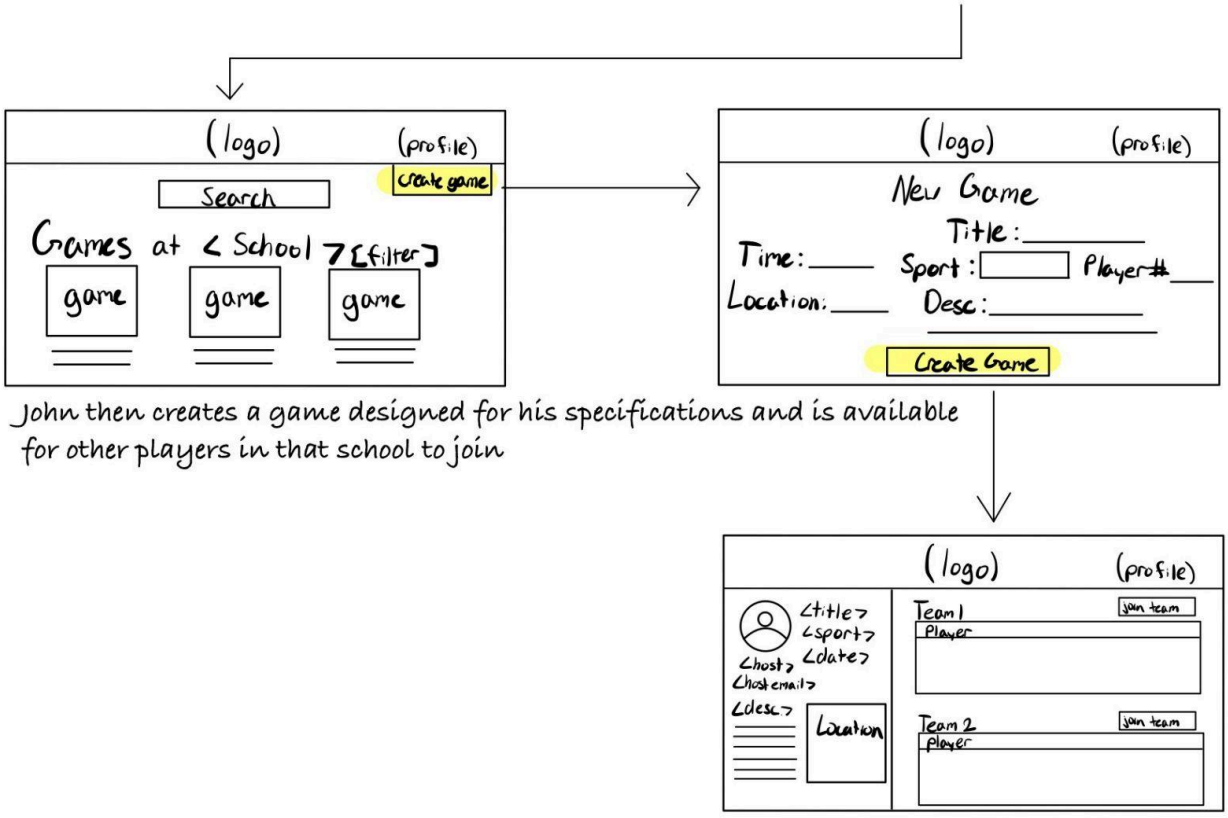

John is greeted by the title screen and clicks login/sign up to create an account




John works for the school, so he fills out the fields accordingly

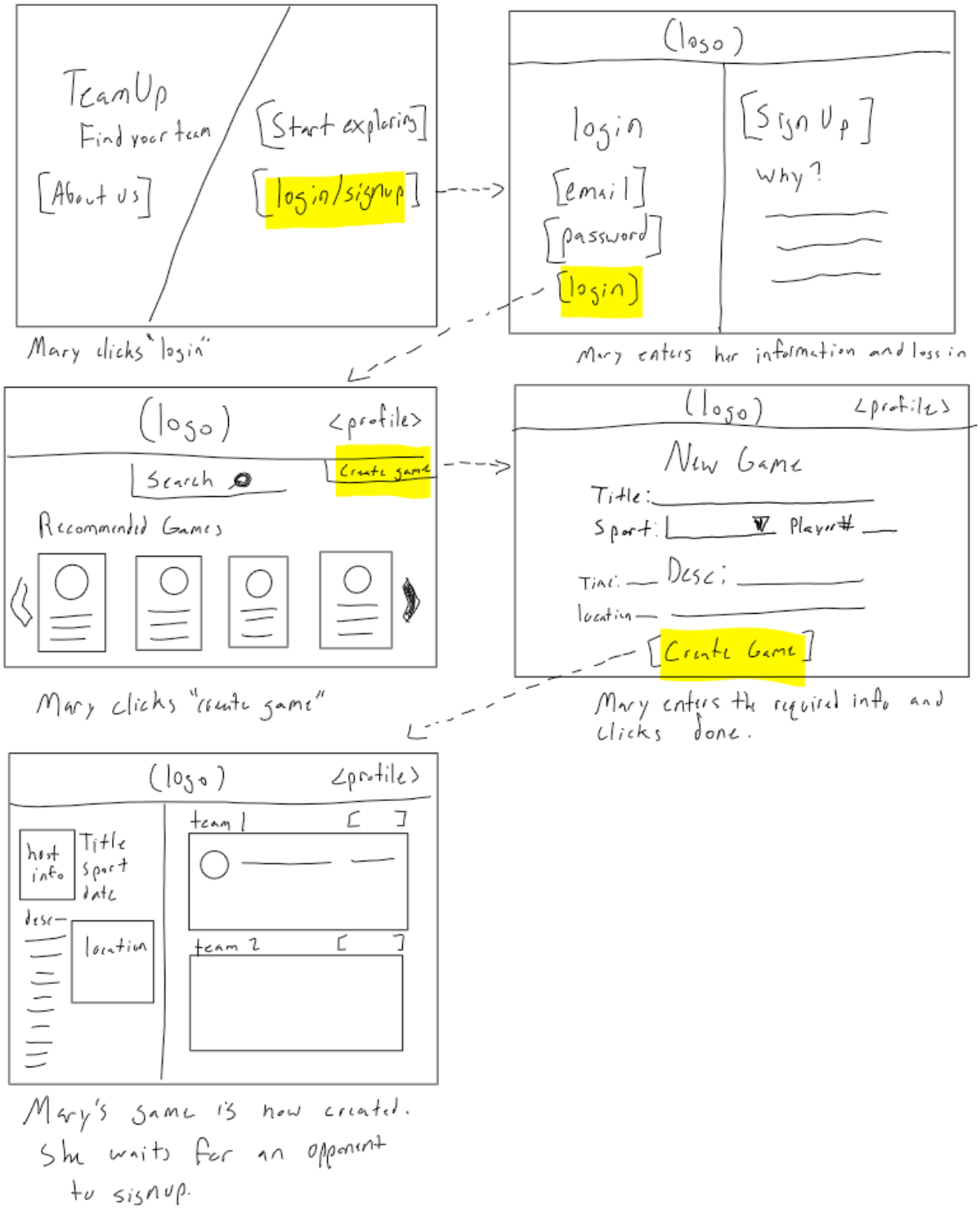
John finds his preferred sport and states his skill level in that sport




(logo)
(profile)
Search
Create game
Games at < School > [filter]
game
game
game
(logo)
(profile)
New Game
Title:
Time:
Sport:
Player#
Location:
Desc:
Create Game
John then creates a game designed for his specifications and is available for other players in that school to join
(logo)
(profile)
<title>
<sport>
<host>
<date>
<host email>
<desc>
Location
Team 1
join team
Player
Team 2
join team
Player

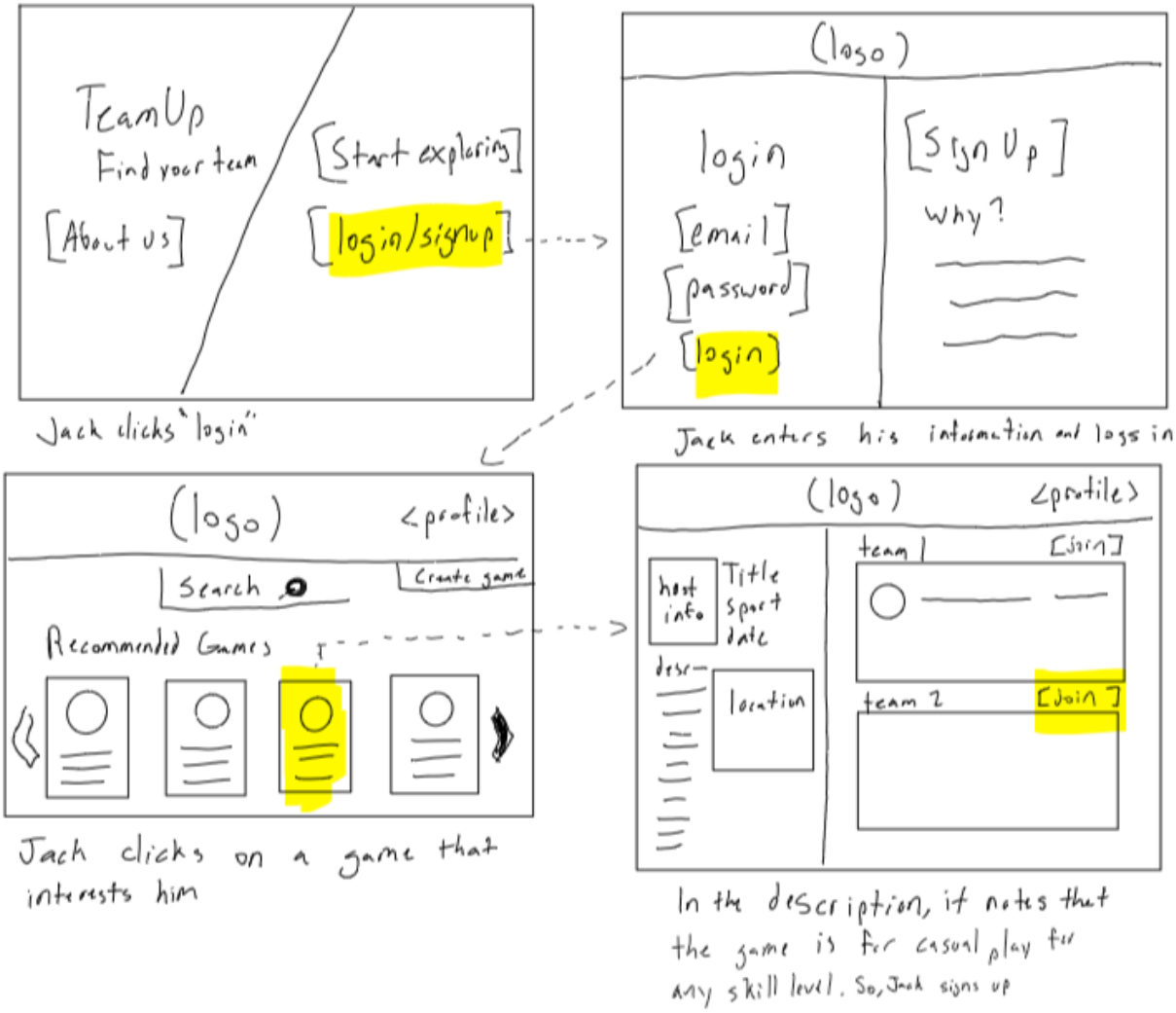
# Use Case 4

- Mary has a tournament coming up
- She wants to get practice in using TeamUp



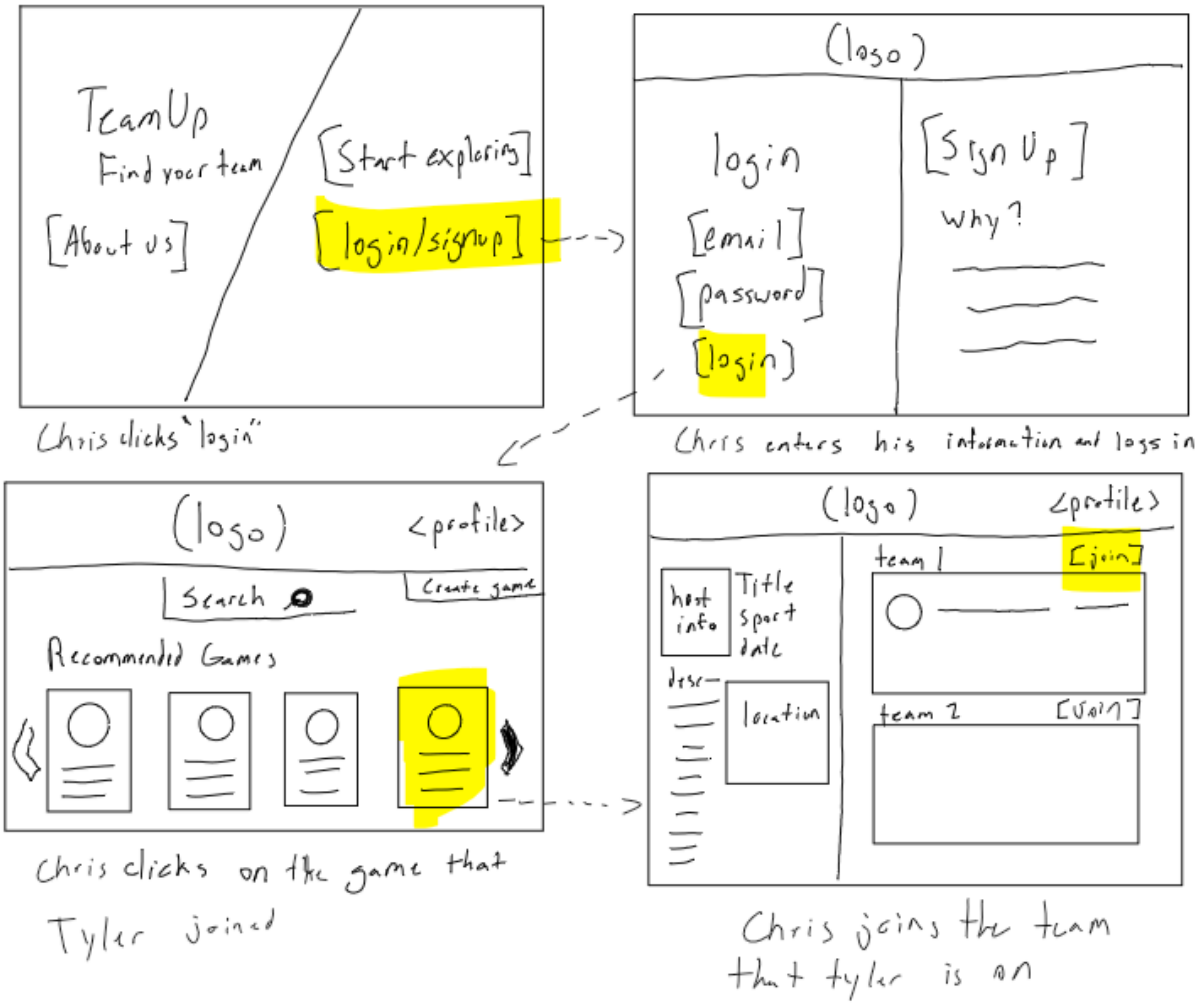
# Use Case 5

- Jack has moved to a new city
- Jack wants to find casual basketball games to play



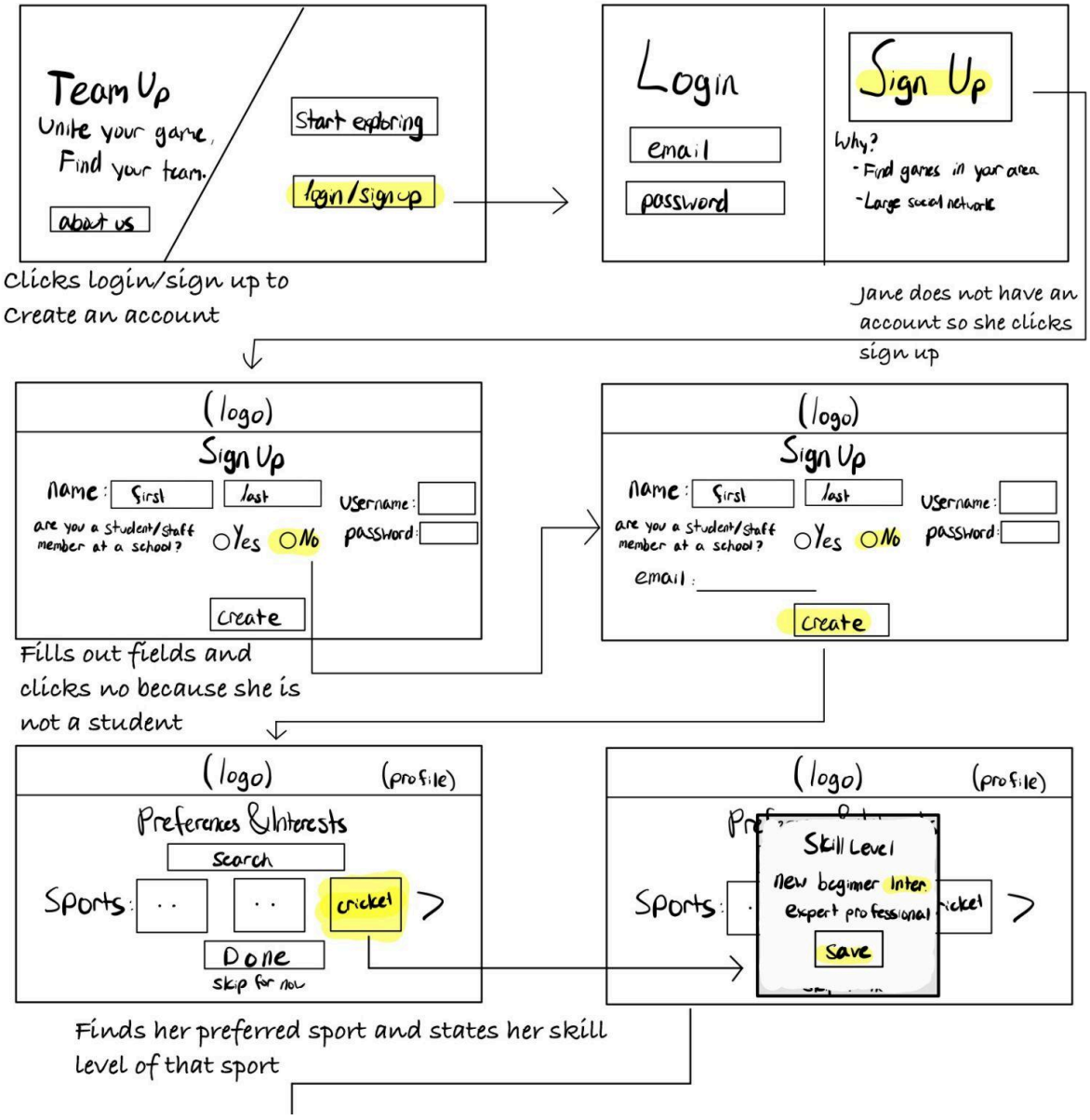
Use case 6

- Tyler has signed up for a game
- His roommate, Chris, also wants to play
- They want to play on the same team



# Use Case 7

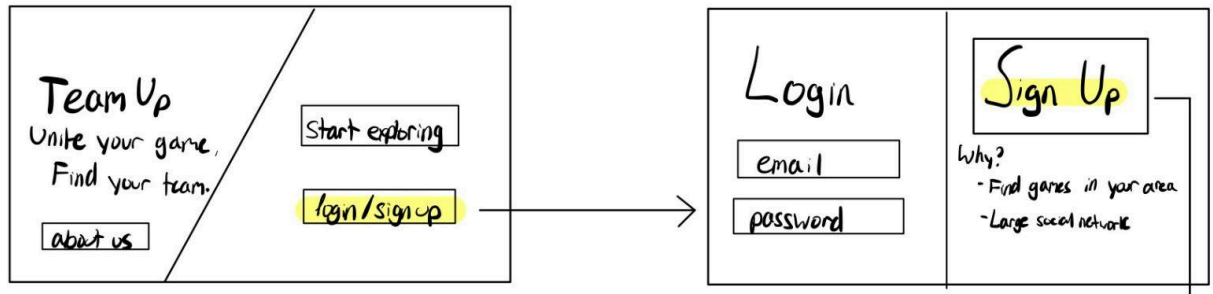
main points:
- Jane moves to San Francisco.
- Jane plays cricket, but cannot find any games nearby.
- Jane comes across TeamUp online and creates an account.
- Jane is now able to find cricket matches in the city.



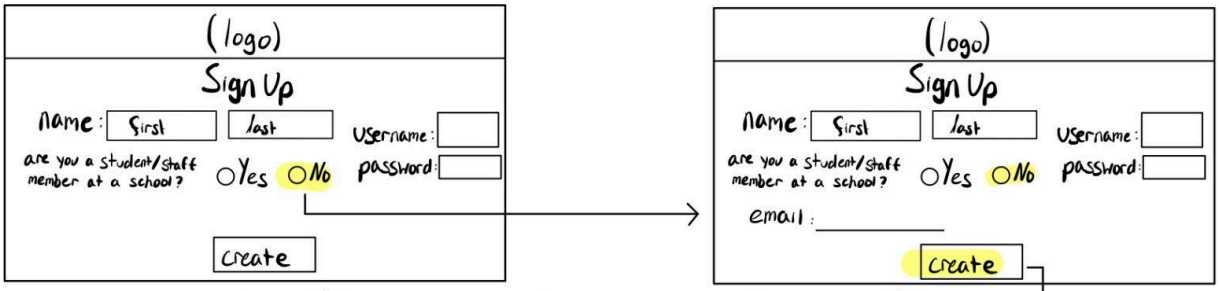
# Use Case 8

main points:

- Trevor and Noah have trouble finding badminton games in their city.
- They discover TeamUp and find that they can set up and find games that suit their availability.
- Trevor and Noah go through the same registration process

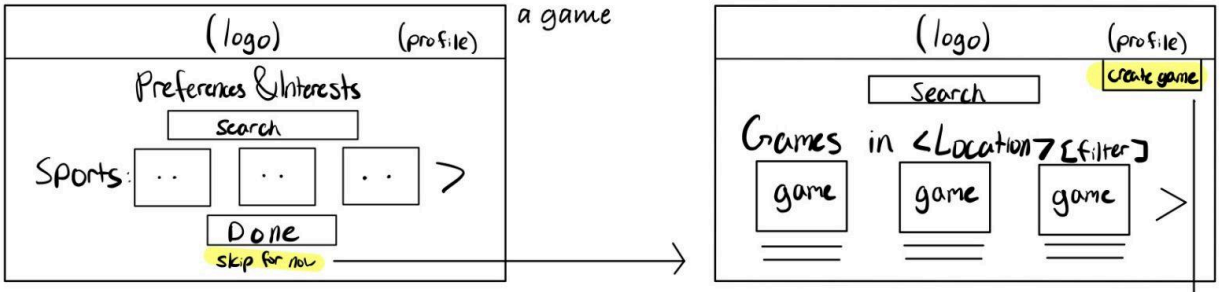


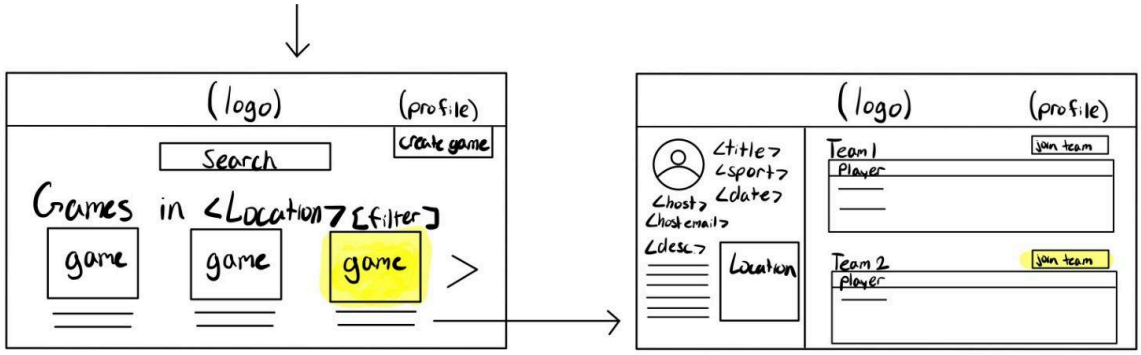

Trevor and Noah click login/sign up to create their accounts



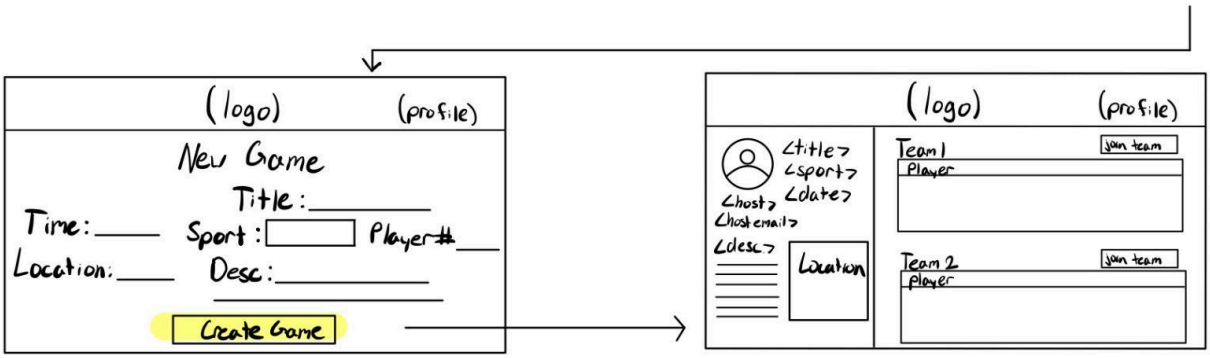

They click "no" because they are not students and fill out the necessary information

One of them clicks the "create game" button to create a game






Finds a game of her choosing in her area and joins a team




The game creator fills out the fields specifying the sport and location and is now visible for others to join

# 4. High-level database architecture and organization

**Functional requirements:**

1. **User**
    - 1.1 a user shall be able to join many games.
    - 1.2 a user shall have only one account with an unique email.
    - 1.3 a user shall host multiple games.
    - 1.4 a user shall have many reviews
    - 1.5 a user shall join many teams
    - 1.6 a user shall have many teammates
    - 1.7 a user shall have many roles
    - 1.8 a user shall have many sport levels.
    - 1.9 a user shall have many sport preferences.
    - 1.10 a user shall have many tokens.
    - 1.11 a user shall write many reviews.

2. **Sport**
    - 2.1 a sport shall be played by many teams.
    - 2.2 a sport shall be chosen by many games.
    - 2.3 a sport shall be chosen as a preference by many users.

3. **Admin**
    - 3.1 an admin is a user.
    - 3.2 an admin shall manage only one school.
    - 3.3 an admin shall be able to create many tournaments.

4. **Game**
    - 4.1 a game shall be hosted by only one user.
    - 4.2 a game shall have multiple players.
    - 4.3 a game shall specify a type of sport.
    - 4.4 a game shall specify the location.
    - 4.5 a game shall be played by at least one team.

5. **Team**
    - 5.1 a team shall have at least one user.
    - 5.2 a team shall specify a type of sport.
    - 5.3 a team shall play many games.

- 5.4 a team shall have only one name.

6. **School**
   - 6.1 a school shall have only one admin.
   - 6.2 a school shall host many tournaments.

7. **Game_Location**
   - 7.1 a game location shall be chosen by many games.
   - 7.2 a game location shall belong to only one city.

8. **Region**
   - 8.1 a region shall have multiple cities.

9. **City**
   - 9.1 a city shall belong to only one region.
   - 9.2 a city shall have multiple game locations.

10. **Review**
    - 10.1 a review shall belong to only one user.
    - 10.2 a review shall be written by only one user.

11. **Token**
    - 11.1 a token shall belong to only one user.

**Entity description:**

1. **User (Strong)**
   - user_id: PK, numeric
   - name: composite (first name, last name), alphanumeric
   - username: alphanumeric
   - email: alphanumeric, email
   - password: alphanumeric
   - dob: multi-value, date
   - Gender: char
   - Phone_number: string
   - created_at: date
   - updated_at: date
   - isEmailVerified: boolean
   - role: enum

User - Admin is a One-to-One relationship.
User - Review is a Many-to-One relationship.
User - Sport is a Many-to-Many relationship.
User - Game is a Many-to-One relationship.
User - Team is a Many-to-Many relationship.

2. **Sport (Strong)**
    - sport_id: PK, numeric
    - sport_name: alphanumeric
    - description: alphanumeric

Sport - User is a Many-to-Many relationship.
Sport - Game is a Many-to-One relationship.
Sport - Team is a Many-to-One relationship.

3. **Admin (Weak)**
    - admin_id: PK, numeric
    - user_id: FK, numeric
    - school_id : FK, numeric

Admin - School is a One-to-One relationship.
Admin - User is a One-to-One relationship.

4. **Game (Weak)**
    - game_id: PK, numeric
    - sport_id: FK, numeric
    - date_time: date
    - number_of_players: Numeric
    - game_location_id: FK, numeric
    - description: alphanumeric
    - fee: alphanumeric
    - gender: alphanumeric
    - age_group: numeric
    - user_id: FK, numeric
    - team_id: FK, numeric

Game - User is a One-to-Many relationship.
Game - Sport is a One-to-Many relationship.
Game - Game Location is a One-to-Many relationship.

Game - Team is a Many-to-Many relationship.

5. **Team (Weak)**
    - team_id: PK, numeric
    - name: alphanumeric
    - sport_id: FK, numeric
    - user_id: FK, numeric

Team - User is a Many-to-Many relationship.
Team - Sport is a One-to-Many relationship.
Team - Game is a Many-to-Many relationship.

6. **School (Strong)**
    - school_id: PK, numeric
    - name: alphanumeric
    - location: alphanumeric

School - Admin is a One-to-One relationship.

7. **Game_Location (Weak)**
    - location_id: PK, numeric
    - name: alphanumeric
    - address: alphanumeric
    - description: alphanumeric
    - parking: alphanumeric
    - fee: alphanumeric
    - map_url: alphanumeric
    - city_id: FK, numeric

Game Location - City is a One-to-Many relationship.
Game Location - Game is a Many-to-One relationship.

8. **Region (Strong)**
    - region_id: PK, numeric

Region - City is a Many-to-One relationship.

9. **City (Weak)**
    - region_id: FK, numeric
    - city_id: PK, numeric

City - Region is a One-to-Many relationship.
City - Game Location is a Many-to-One relationship.

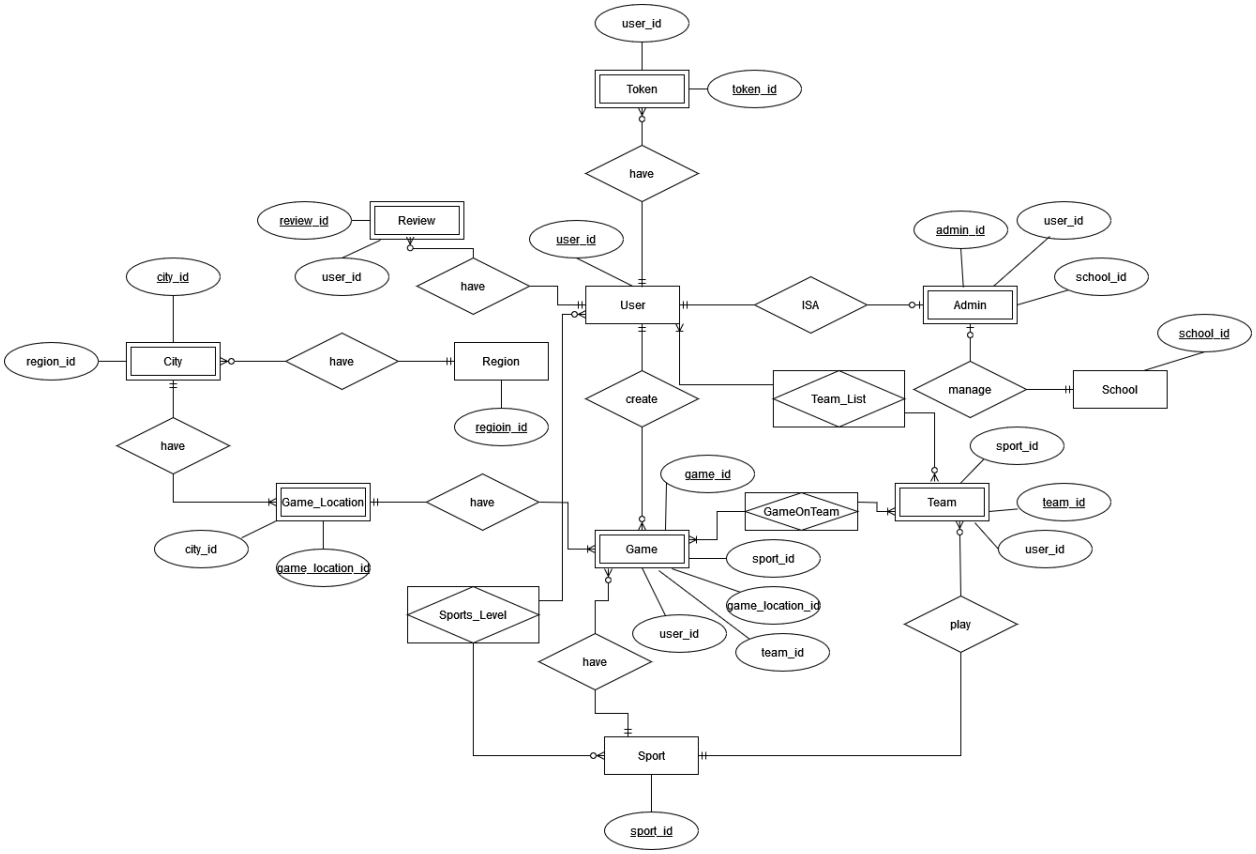
10. **Review (Weak)**
    - review_id: PK, numeric
    - user_id: FK, numeric
    - rating: alphanumeric
    - description: alphanumeric

Review - User is a One-to-Many relationship.

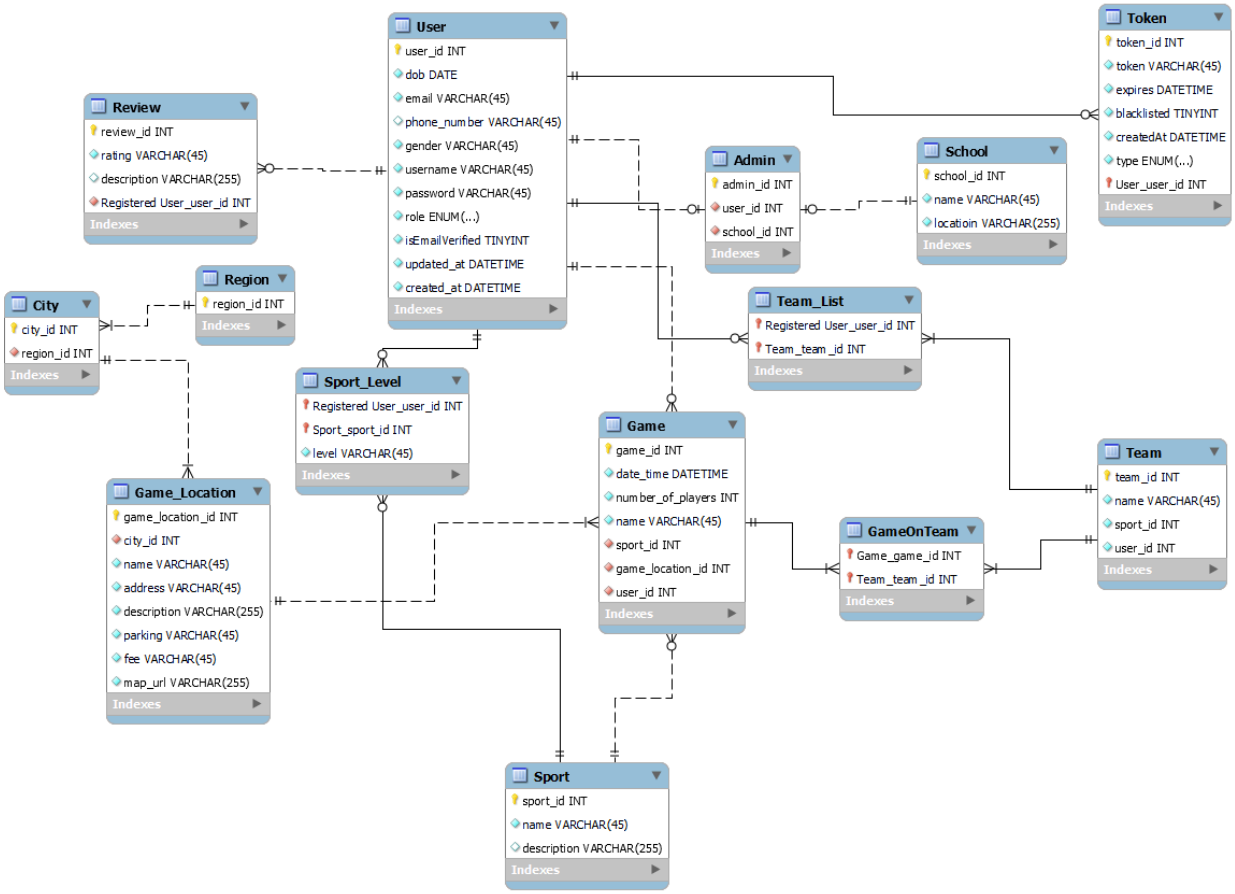
11. **Token (Weak)**
    - token_id: PK
    - user_id: FK
    - token: alphanumeric
    - type: enum
    - expires: date
    - blacklisted: boolean
    - createdAt: date

Token - User is a One-to-Many relationship.

**ERD:**


user_id
Token
token_id
have
review_id
Review
user_id
have
user_id
User
ISA
admin_id
user_id
Admin
school_id
city_id
region_id
City
have
Region
regioin_id
have
create
Team_List
manage
School
school_id
sport_id
game_id
Game_Location
have
GameOnTeam
Team
team_id
city_id
game_location_id
Game
sport_id
user_id
Sports_Level
game_location_id
user_id
team_id
play
have
Sport
sport_id

**EER:**




• Media storage: **We will use Amazon s3 (Amazon Simple Storage Service). It provides object storage through a web service interface. It's low cost and it has unlimited storage capacity. We decided to use it because it has high scalability, high durability, and high availability. It's great because since it has no capacity limit, we don't have to worry about data capacity we have left. Also, it's guaranteed by Amazon that its durability is 99.9%, and it's the same for availability for every storage class.**

# 5. High-Level APIs and Main Algorithms

This part of the documentation contains major building blocks of our web app's communication and decision-making processes. First, we will outline the main APIs we plan to create. These will establish a connection between our react backend and expressJS backend, making sure that the transactions happening on the frontend gets reflected on the backend properly.

## High-Level API Specs:

### Signup

**Endpoint Description:** This endpoint allows new users to create an account on the platform. Users are required to provide a valid email address and a password. The system will then process this information, perform necessary validations (such as checking for the uniqueness of the email), and create a new user account if all criteria are met. Upon successful account creation, the user may receive a confirmation email or message acknowledging their new account.

### Login

**Endpoint Description:** This endpoint facilitates user authentication on the platform. Users must submit their registered email and password. The system verifies these credentials against the stored user data. If the credentials are correct and the user is authenticated successfully, the system grants access to the user, often generating a session token or authorization token that can be used for subsequent requests to authenticate the user.

### View Games

**Endpoint Description:** This endpoint allows authenticated users to discover games that are happening close to their current location. Users may need to provide location information (either explicitly or implicitly through device location permissions) for the system to determine which games are in close proximity. The system then retrieves and returns a list of nearby games, including details such as game type, location, time, and possibly the number of participants or slots available.

### Search Game

**Endpoint Description:** This endpoint allows authenticated users to search for games based on multiple filters, including sport type, location, time, and player skill level. Users can submit a

search request with any combination of these filters to find games that match their preferences. The system will return a list of games that meet the criteria, including details like game type, location, time, number of participants needed, and skill level requirements.

### Create Game

**Endpoint Description:** Authenticated users can use this endpoint to create new games. Users must provide details such as the sport type, location, time, number of players required, and any specific skill level requirements. The system processes this information to create a new game entry in the database, which becomes discoverable to other users through the "Search Games" endpoint. Game creators have the ability to manage their game, including inviting friends to join.

### Game Listing Details

**Endpoint Description:** When users view or search for games, each game listing will include comprehensive details such as the facility location, player profiles of participants, and facility game rules. This ensures that users have all necessary information to make informed decisions about joining or creating games.

### Get User Profile

**Endpoint Description:** This endpoint retrieves the profile information of a specific user on the platform. By supplying a user's unique identifier (userId), the system fetches and returns the user's profile data, including email address, skill level, preferred sports, and historical data on games they have joined. This functionality allows users to review their own profile details or enables other users to view certain information about potential game participants, thereby enhancing user interaction and engagement within the platform.

### Get Joined Games

**Endpoint Description:** The "Get Joined Games" endpoint is designed to provide users with a list of all the games they have joined. By passing the user's unique identifier (userId), the system will return a detailed list of games, showcasing information such as the type of sport, location, time of the game, and the user's role or status in those games. This feature is crucial for users to manage their schedules, keep track of upcoming games, and reflect on past participations.

### Join Game

**Endpoint Description:** This endpoint facilitates users in joining an existing game. Users are required to submit the unique game ID of the game they wish to join along

with their user ID. The system then validates the request by checking for available slots, matching the user's skill level with the game's requirements, and ensuring the game's capacity is not exceeded. Upon successful validation, the user is added to the game's participant list, and a confirmation is returned. This process is designed to ensure that games are accessible and inclusive, allowing users to easily engage with activities of their interest.

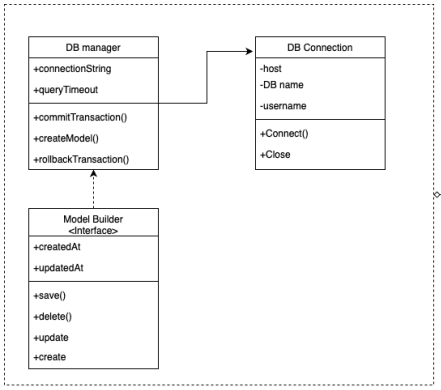
## New Framework:

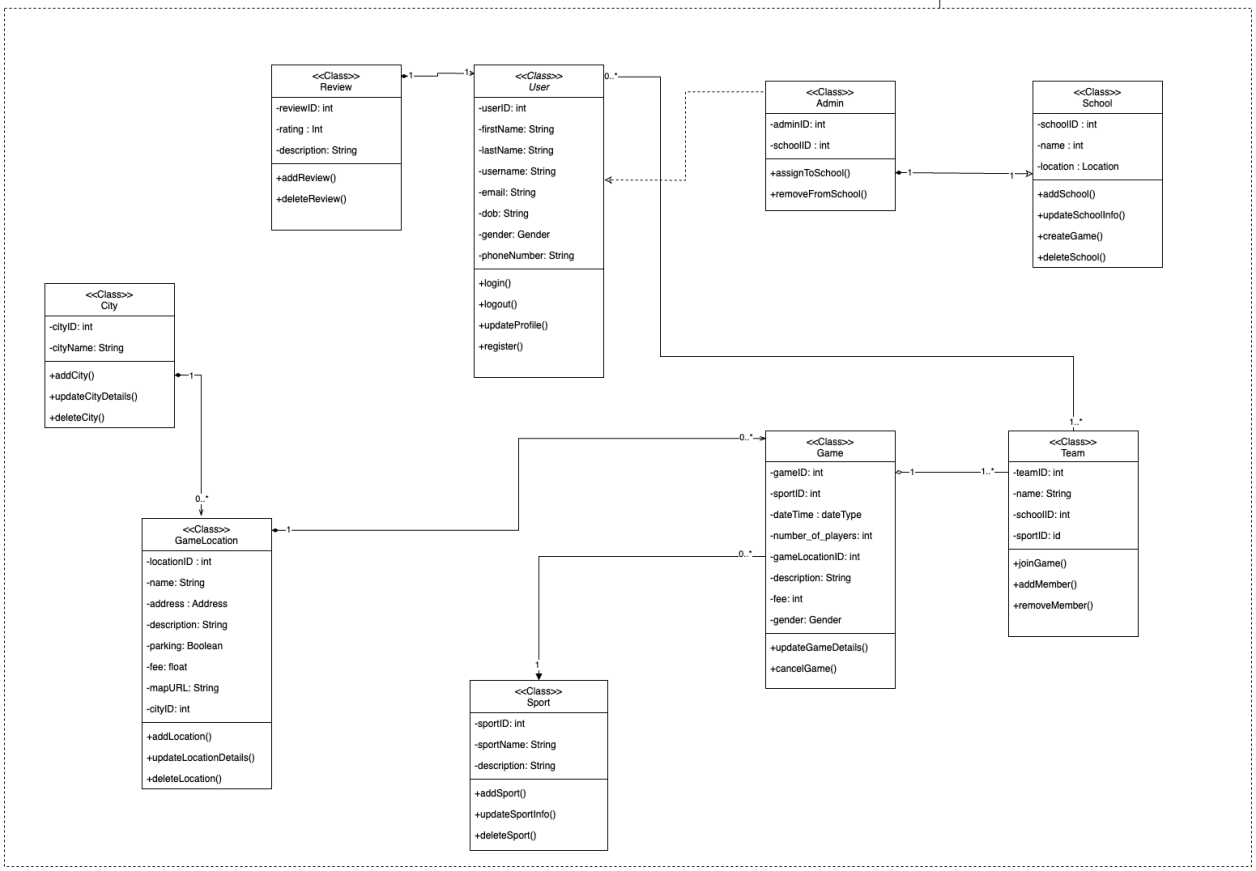
One new tool that we have decided to use is Prisma, because it makes working with databases much easier and more efficient. Prisma is great because it lets us handle databases in a simpler, safer, and more developer-friendly way. We chose it because we want our development process to be strong, quick, and productive.

With Prisma, setting up database structures and running queries becomes a lot simpler. It supports many different databases, which is really helpful. It saves us time by taking care of mundane tasks and helps reduce mistakes. This is super important for keeping our work moving fast. Plus, Prisma's focus on safety and its organized way of managing database changes make us feel good about our app's data being secure and able to grow. By going with Prisma, we're not just selecting a tool; we're enhancing how we manage data in our application.

# 6. System Design

## UML Class diagrams



This UML class diagram is built for our backend development process by implementing a clear object-oriented paradigm. It serves as a blueprint that not only guides the creation of our system's components but also ensures that each part plays a defined role. The primary goal is to minimize code redundancy, which is achieved by identifying and encapsulating common behaviors and data structures within classes and interfaces.

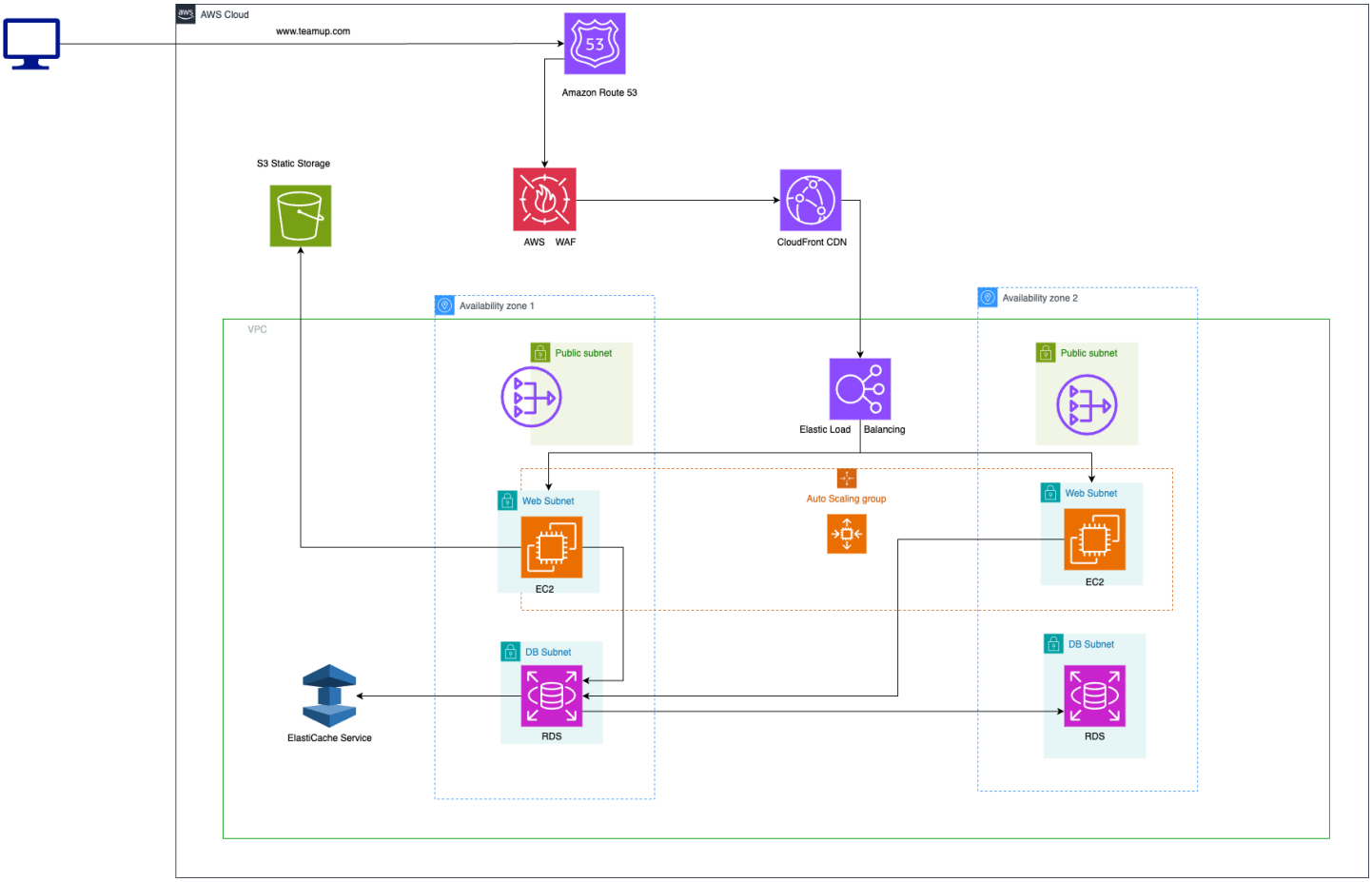
We've adopted design patterns that are conducive to  modularity. For instance, the 'DB manager' class centralizes data management functions, offering a single point of interaction with the database through a well-defined interface. This encapsulation simplifies maintenance and future upgrades to data handling mechanisms.

The use of interfaces, such as 'Model Builder', delineates a contract for creating and managing persistent entities. This pattern allows for flexible implementation strategies without affecting the clients of these interfaces. By adhering to these interfaces, the system can evolve over time without necessitating changes in the classes that rely on them.

The adapter and bridge connectors, represented by 'AdapterConnector' and 'BridgeConnector', demonstrate our commitment to an adaptable system architecture. They enable the integration of components with incompatible interfaces and decouple abstraction from implementation, respectively. This will be particularly beneficial when incorporating third-party services or when multiple implementations of a component are required.

Furthermore, the systematic classification of domain entities, like 'User', 'Review', 'City', and 'Team', encapsulates both data and behavior relevant to each concept. This organization clarifies the system's architecture, making it more navigable for new developers and enhancing collective understanding. Each class serves a specific purpose, reducing the temptation to intermingle responsibilities that can lead to bloated, less maintainable code.

Scalability Diagram:




- When a user types in the web address (like www.teamup.com), Amazon Route 53 acts as the signpost, directing them to the right server.

- Amazon CloudFront, a content delivery network, steps in to serve the user's request by fetching the content from the nearest location to the user. This ensures the data travels a shorter distance, speeding up the delivery of the website's static assets like images or videos.

- AWS WAF, integrated with CloudFront, functions as a shield, protecting the website from malicious attacks by filtering traffic based on set rules.

- The user's request then reaches the Elastic Load Balancer, which is like a revolving door that efficiently manages incoming traffic, ensuring no single server gets overwhelmed.

- If the user's request involves data that changes frequently, it is routed to EC2 instances within an Auto Scaling group across multiple Availability Zones. This Auto Scaling adjusts the number of EC2 instances in response to the demand, making sure the application can handle the load without manual intervention.

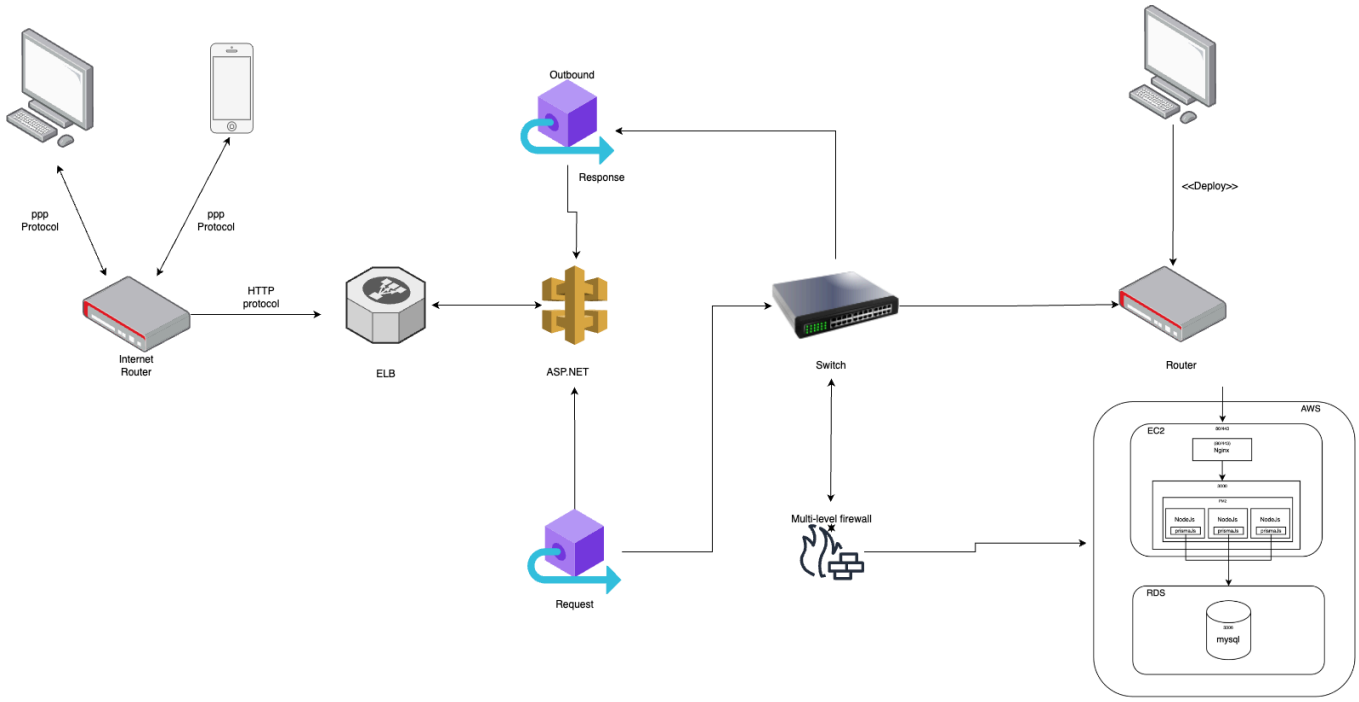
- If the request is for data that doesn't change often, Amazon ElastiCache steps in to provide this data quickly, as it's a temporary storage that remembers frequently accessed information.

- For storing and managing the application's data, Amazon RDS is used, which replicates the data across multiple locations (Availability Zones) to ensure that even if one database server fails, another can take over without loss of data or service.

- All of these components operate within a VPC, which is like a gated community ensuring secure communication between the resources, and Security Groups act as the rules for who can talk to who within this community.

- Finally, for storing static resources and backups, Amazon S3 is used as a vast, secure storage facility, where data is stored safely and can be accessed or restored as needed.

# 7. High Level Application Network and Deployment Design




# 8. Identify actual key risks for your project at this time

1. **Skills risks (do you have the right skills)**

Risk: The team may lack expertise in certain technologies that are required for the app development. The team may not be aware of these technologies because we are still early in development. Unfamiliar technologies may pop up when implementing features such as real time notifications.

Resolution: Research ahead of time what tools will be needed for implementing a lot of the features that our app will need. If there is a lack of knowledge, plan to train accordingly to ensure that the team can use these tools effectively.

2. **Schedule risks (can you make it given what you committed and the resources)**

Risk: Unrealistic goals and limited resources can cause the development of our app to be delayed.
Resolution: Have an extremely thorough project planning session with input from the entire team.

3. **Technical risks (any technical unknowns to solve)**

Risk: Unanticipated technical challenges can very much arise during implementation of many features in our app.
Resolution: Before jumping into it, we will make prototypes for essential components to identify and address potential issues early on. We will also be sure to be communicative with each other about any technical concerns.

4. **Teamwork risks (any issues related to teamwork)**

Risk: Poor communication within the team can have a negative effect on collaboration and productivity.
Resolution: Team will work towards creating a culture that welcomes open communication and collaboration. We will all commit to our regularly conducted team meetings to discuss progress and address any concerns.

5. **Legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright)**

Risk: May run into issues while using specific content or software.
Resolution: Perform a thorough review of licensing requirements for any software and content used. Use images with proper licensing agreements. Utilize websites with images that aren't copyrighted or are available under licenses allowing their use such as Unsplash or PicJumbo.

# 9. Project management

Getting through this milestone felt a bit tougher than usual. We had to tackle some tasks that our professor didn't lay out for us, which meant we had to get creative and figure things out as a team. For organizing our work, we've been using ClickUp. It's been super helpful, but one thing we've noticed is that we're not always the best at sticking to our deadlines. We definitely need to tighten up on that.

One of the highlights for me has been the fantastic support from our frontend lead. They've really taken charge of managing the frontend tasks, which has been a huge relief. It's allowed me to spend more time focusing on the backend, without worrying too much about what's happening on the frontend side. It's made a big difference in how smoothly things have run.

Looking ahead to the next milestone, I'm thinking we should introduce weekly tasks with stricter deadlines. It seems like a good way to keep us on track and make sure we're moving forward steadily. This experience has been a learning curve, but I'm excited to see how these changes will help us work even better as a team.

## 10. Detailed list of contributions

| | | |
|---|---|---|
| Juan Estrada | 9 | -Submit Milestone 2 & create the workflow for Milestone 2<br>-Registration API<br>-Create Game API<br>-[BE - API] Get User Profile<br>-[BE - API] View Games<br>-UML<br>-Scalability Diagram |

| | | |
|---|---|---|
| Kotaro Iwanaga | 9 | -Database Architecture<br>-EER<br>-ERD<br>-Prisma Model |
| Cole Chiodo | 9 | -Create black and white mockups/story boards for main cases(4,5,6)<br>-Signup page<br>-Data Definitions |
| Martin Pham | 9 | -Risk Management Team<br>-Create black and white mockups/story boards for main cases(3,4)<br>-Landing page<br>-Home page |
| Jaycee Lorenzo | 9 | -Lead frontend tasks<br>-Create black and white mockups/story boards for main cases(1,2)<br>-Login page<br>-Home page |
| Areeb abbasi | 9 | -Deploy app<br>-API and Algorithm<br>-[BE - API] Search Games<br>-Main Data Items and Entities<br>-Login API<br>- High Level Application Network and Deployment Design |

SW Engineering CSC648-848-05 Spring 2024

Application Title & Name : Teamup

Team : 5

| Student Name | SFSU Email | GitHub | Discord | Role |
|---|---|---|---|---|
| Juan Estrada | jestradazuluaga@sfsu.edu | jjestrada2 | juanjosee | Team-lead |
| Areeb Abbasi | aabbasi@sfsu.edu | areeeeb | _xertz | Backend-lead |
| Cole Chiodo | cchiodo@sfsu.edu | colechiodo | colechiodo | Docs-editor |
| Jaycee Lorenzo | jlorenzo3@sfsu.edu | jclorenz0 | __jaycee | Frontend-lead |
| Martin Pham | mpham8@sfsu.edu | mar10fam | marnoki | Github-master |
| Kotaro Iwanaga | kiwanaga@sfsu.edu | iamkotaaa | kotaro8448 | Database-admin |

Milestone 3

4/25/2024

| Version # | Submission Date |
|---|---|
| M3V2 | 05/14/2024 |
| M3V1 | 04/25/2024 |
| M2V2 | 04/18/2024 |
| M2V1 | 04/04/2024 |
| M1V2 | 03/21/2024 |
| M1V1 | 03/01/2024 |

# 1. Data Definitions

## 1. User

a. Definition: Represents anyone who interacts with the app, including players and school administrators
b. Usage: Users can browse games, join or create games, and participate in tournaments. School administrators have additional privileges to create and manage tournaments.
c. Attributes: User ID, Name, Email, Password, Profile Picture (JPG, 320x320px), Skill Level, Sports Preferences, Availability.

## 2. Game

a. Definition: A scheduled sports activity that users can join or create.
b. Usage: Allows users to find, join, or post games matching their interests and schedules.
c. Attributes: Game ID, Sport Type, Location, Date & Time, Required Number of Players, Skill Level Preference, Equipment Details, Organizer (User ID).

## 3. Team

a. Definition: A group of users who join together to play a sport, either for casual games or within a tournament.
b. Usage: Facilitates team formation for both casual play and competitive tournaments.
c. Attributes: Team ID, Team Name, Members (List of User IDs), Sport Type, Skill Level.

## 4. Tournament

a. Definition: A competitive event organized by schools or universities, involving multiple games and teams.
b. Usage: Schools can create tournaments, and students can register as solo players, groups, or teams.
c. Attributes: Tournament ID, Name, Sport Type, Date & Time, Location, Registration Details, Tournament Format, Brackets, Standings.

## 5. Game Location

a. Definition: Physical place where games and tournaments are held.
b. Usage: Users can select locations for their games or view where tournament matches are taking place.
c. Attributes: Location ID, Name, Address, Facility Details, Parking Information.

6. Profile
    a. Definition: Detailed information about a user, including their sports preferences and skill levels.
    b. Usage: Helps in matching users with appropriate games and teams.
    c. Attributes: User ID, Skill Levels (per sport), Sports Preferences, Biography, Contact Information.
7. Admin
    a. Definition: A user who is associated with one school who can create tournaments.
    b. Usage: Admins are approved users with higher privileges. They are able to create tournaments for the school they are associated with.
    c. Attributes: Admin ID, User ID, School ID
8. School
    a. Definition: Location that can host Tournaments.
    b. Usage: Locations that Admins can host Tournaments at.
    c. Attributes: School ID, Name, Location.
9. Sport
    a. Definition: The activity that will be played during a game.
    b. Usage: When creating a game, a sport is chosen that will be played. Users can also define their sport preferences.
    c. Attributes: Sport ID, Name, Description.
10. Review
    a. Definition: A rating given from one user to another.
    b. Usage: Users can rate other users based on their performance or actions after they have played a game together.
    c. Attributes: Review ID, Rating, Description, User ID.
11. Sport Level
    a. Definition: A level of expertise
    b. Usage: Users can filter games based on the level of expertise of a specific game.
    c. Attributes: User ID, Sport ID, Level.

## User Types and Privileges

● Player: Regular users who can browse and join games, create games, and register for tournaments. They can also be part of or form teams.
● School Administrator: Users with the authority to create and manage

tournaments on behalf of their institution. They have additional access to tournament management tools.

## Registration Info

- Upon account registration, the user must provide the following items of information:
    - Full Name, Username, Email, Password, Date of Birth, Gender, Phone Number.

## Usage in Documentation and Development

- These terms and their definitions will be used consistently across all project documentation, user interfaces, software components, and database designs.
- The division of user types into Players and School Administrators informs the app's functionality and access control mechanisms.
- The attributes listed for each entity provide a high-level overview of the data model and serve as a guide for database design and API development.

# 2. Prioritized Functional Requirements

## Priority 1:

### User:

1. Users shall be able to search for games based on sport type.
2. Users shall be able to search for games based on location.
3. Users shall be able to search for games based on time.
4. Users shall be able to join a team in a game.
5. Users shall be able to create new games.
6. Users shall be able to specify the sport for a game.
7. Users shall be able to specify the location of a game.
8. Users shall be able to specify the time of a game.
9. Users shall be able to specify the number of players needed for a game.
10. Users shall be allowed to update their account profile picture.
11. Users shall be allowed to update their account biography description.
12. Users shall have the option to manually log out from their accounts.
13. Users shall have the option to verify their accounts through email.
14. Users shall be able to recover their password through email.
15. Users shall be able to detach from a game.
16. Users shall be able to search all their joined games.
17. Users shall be able to search all their hosted games.

### Game listings:

18. Game listings shall be able to provide a facility location map.
19. Game listings shall be able to provide the player's username.
20. Game listings shall be able to provide player’s pictures.
21. Game listings shall be able to provide the player’s link to their biography.
22. Game listings shall be able to provide facility game rules.
23. Game listings shall be able to provide the name of the organizer.
24. Game listings shall be able to provide the name of the game.
25. Game listings shall be able to provide the day and time of the game.
26. Game listings shall be able to provide the contact information of the organizer.
27. Game listings shall be able to provide a short description of the game.
28. Game listings shall be able to provide the empty slots of the game.

### Game Location:

29. Game locations shall be able to show address and facility details.
30. Game locations shall be able to provide parking information.

31. Game locations shall be able to provide users with reviews.

## Sport:

32. Users shall be able to choose sports preferences
33. Users shall be able to select their skill level in a specific sport.

## Team:

34. Teams shall be able to be formed in a game listing.
35. Teams shall be restricted by a specific number of players in a game.

## Profile:

36. Profile preferences shall be able to be edited by the user.
37. Profile pictures shall be able to be uploaded to user's bio(JPG, 320x320px).
38. Profile email shall be able to be edited by the user.
39. Profile phone number shall be able to be edited by the user.
40. Profile gender shall be able to be edited by the user.
41. Profile birthday shall be able to be edited by the user.
42. Profile password shall be able to be edited by the user.

## Review:

43. Users shall be able to rate other players.
44. Users shall be able to write reviews for other players.
45. Users shall be able to rate facilities for game locations.
46. Users shall be able to write reviews for s for game locations.

## Priority 2:

### User:

47. Users shall be able to search for games based on player skill level.
48. Users shall earn badges for achievements.
49. Users shall receive confirmation emails containing game details.
50. Users shall receive notifications containing game participation instructions.
51. Users shall receive reminders before the game start date.
52. Team members shall be able to communicate to coordinate logistical issues related to games.
53. Participants shall receive notifications of their game schedules, including date, time, opponent, and location.
54. The app shall recognize the outstanding performances of top participants.
55. Users shall have the opportunity to provide feedback and evaluations on the game.
56. Users shall have privacy settings to control who can see their profile and game activity.
57. Users shall be able to use optional two-factor authentication (2FA).
58. Users shall see leaderboards for various metrics like most games played.

### Profile:

59. Users shall be able to manage their contact information.
60. Users shall be able to view and update their skill levels for each sport.

### Tournament:

61. Admins shall be able to create and manage tournaments.
62. Tournaments shall allow users to register as teams.

### Admin:

63. Admins shall be able to create tournaments for their associated schools.

### School:

64. Schools shall be able to host tournaments.
65. Schools shall be able to manage their associated admins.

### Team:

66. Users shall be able to form teams for games.
67. Teams shall be able to manage their members.
68. Teams shall be able to schedule games.

### Sport:

69. Users shall be able to filter games based on the level of expertise required.

## Priority 3:

### Game listings:

70. Games shall be able to provide suggestions for indoor locations in case of predicted bad weather.
71. Games shall include options for renting necessary sports equipment.
72. Games shall support calendar integrations for managing team schedules.
73. Games shall include weather forecasts for the scheduled time and location.

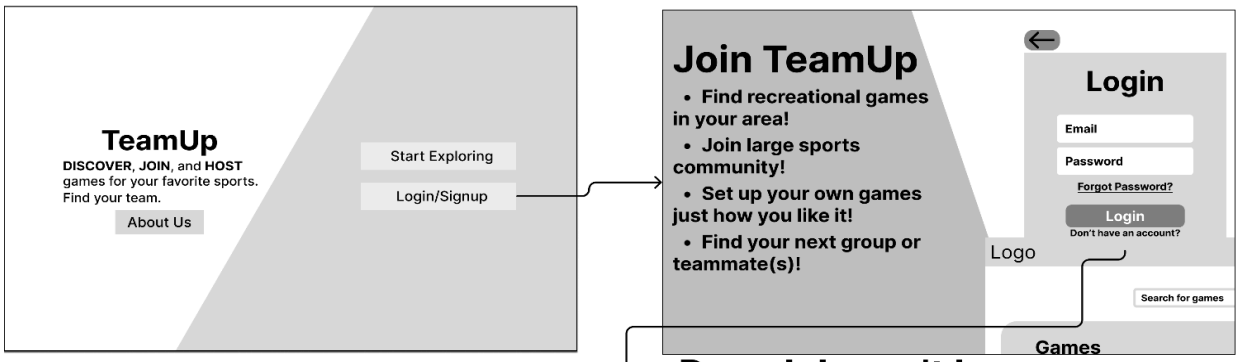
### Tournament:

74. Tournaments shall offer options to livestream games.
75. Tournaments shall generate game schedules based on the number of teams, available time slots, and tournament format.
76. Admins shall have the ability to update or change the tournament schedules.
77. Tournaments shall enforce a code of conduct policy for game participants.
78. Violations of the code of conduct shall be reported.
79. Admins shall be able to manage team rosters, set up team events, and send out team announcements.

# 3. Wireframes Based on your Mockups/Storyboards

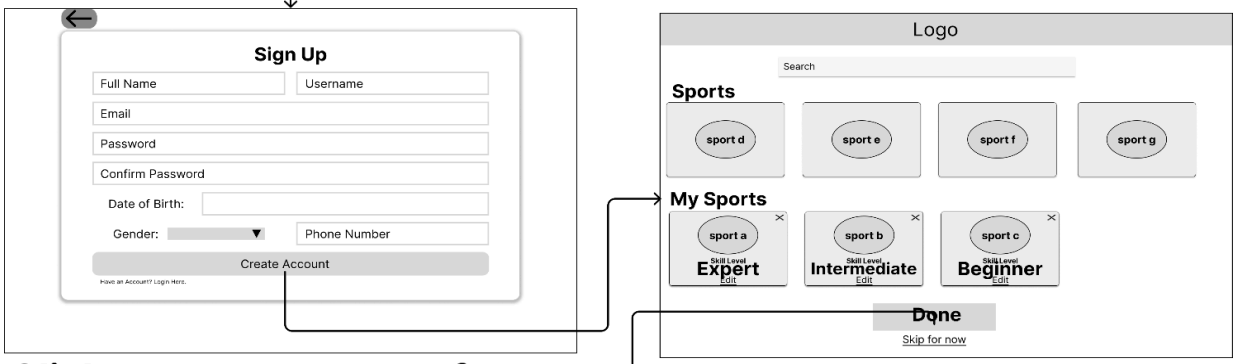
## Use Case 1

- **Darryl wants a way to meet friends at SFSU**
- **He meets people by signing up with TeamUp and joining games**



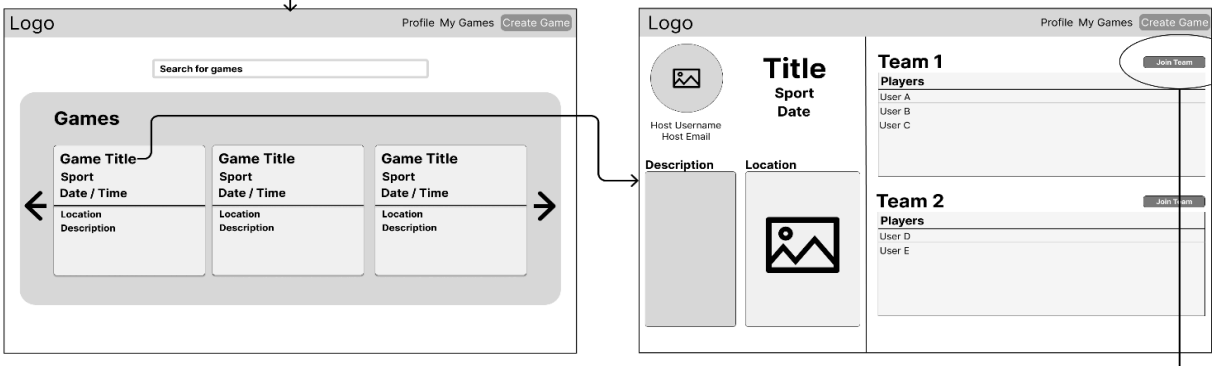

**Darryl clicks login/signup**

**Darryl doesn't have an account so goes to sign up**




**Clicks create account after entering info**

**Selects his favorite sports and continues to home page**

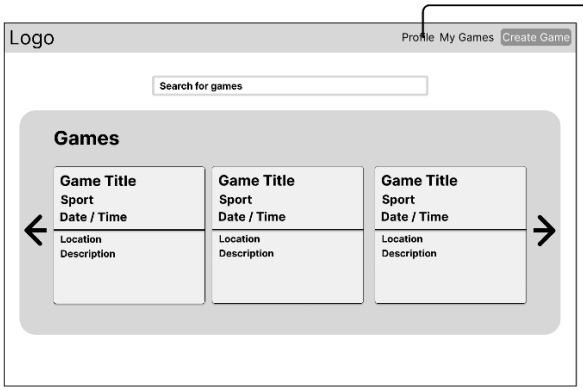



**Clicks on a game hosted by other SFSU students**

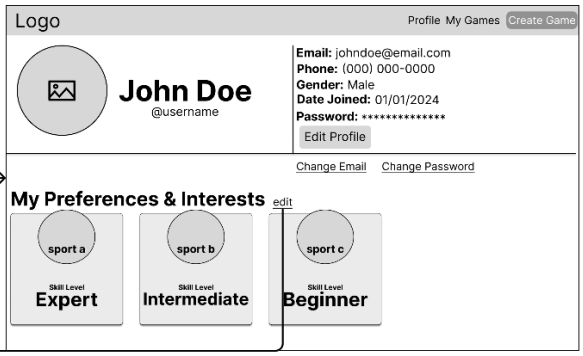
**Joins a team**

## Use Case 2

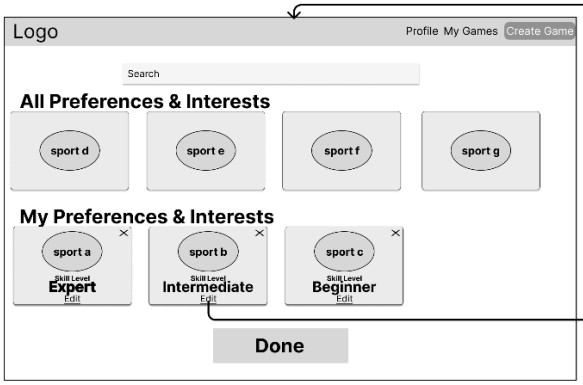
- **Alex is an existing member on TeamUp**
- **He believes it is necessary to adjust his skill level**
- **After adjusting, he has a much more enjoyable experience playing with other players with similar skill level**




**Alex goes to his profile**

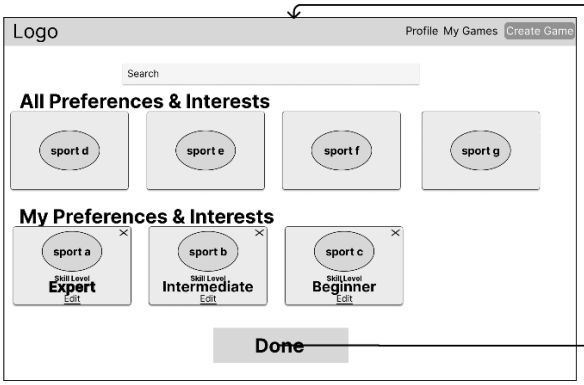



**Clicks edit to change his skill level**

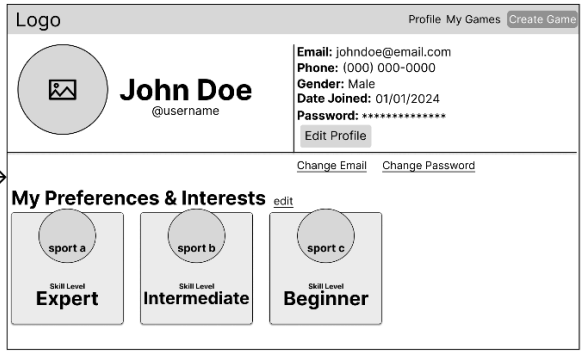



**Clicks edit on the sport he wants to change**


**Selects beginner and saves**

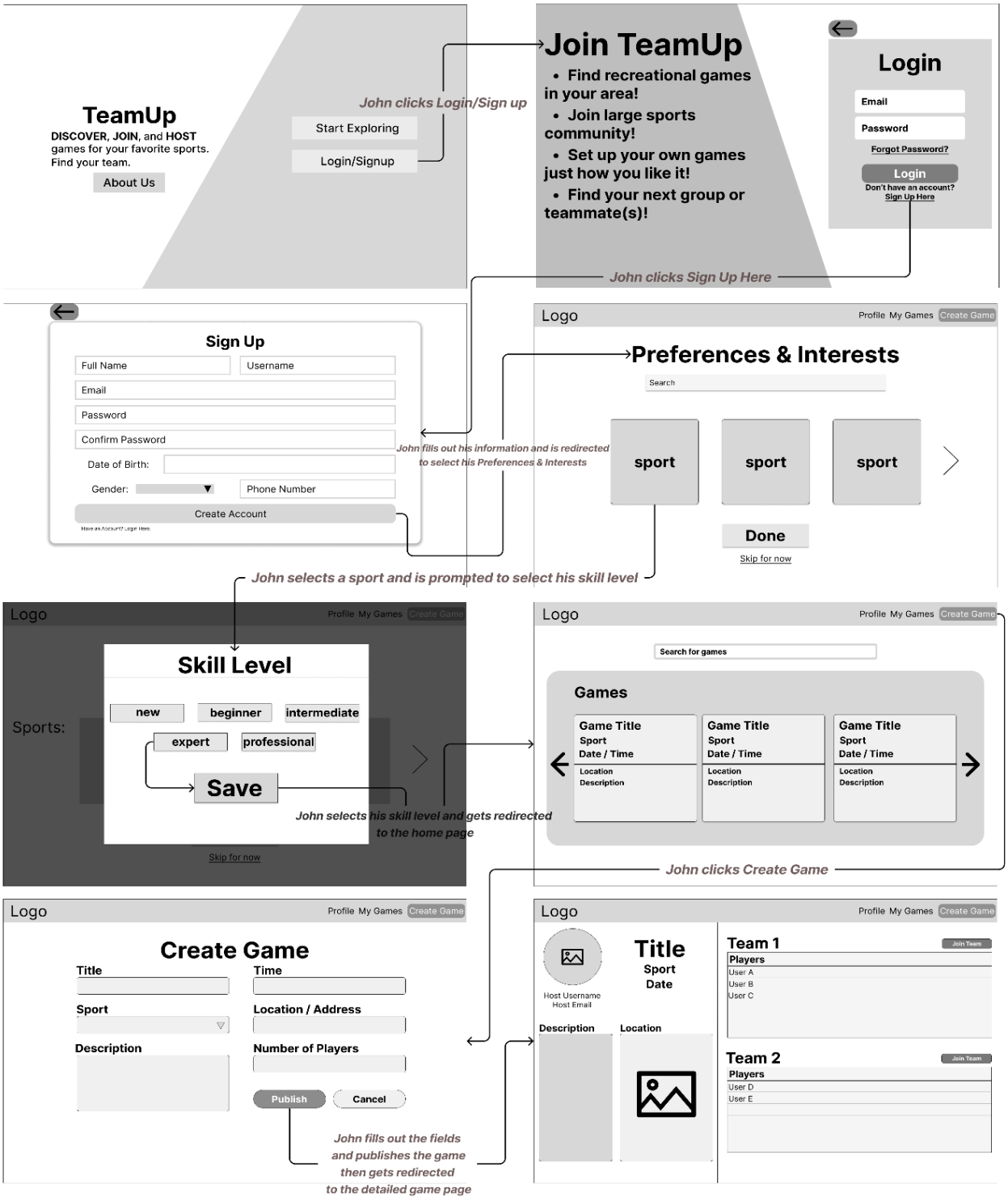



**Clicks done and sees changes back in his profile page**



# Use Case 3

- **John wants to set up basketball games with other high skill level players**
- **John gets introduced to TeamUp**
- **John Registers and creates a game cater to more experienced players**

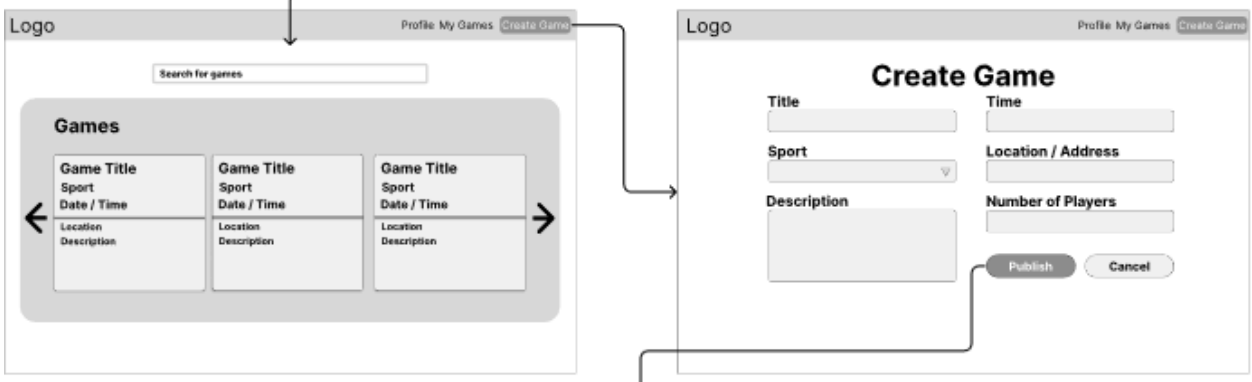


## Use Case 4

- **Mary has a Tennis Tournament soon**
- **She wants to get extra practice in using TeamUp**

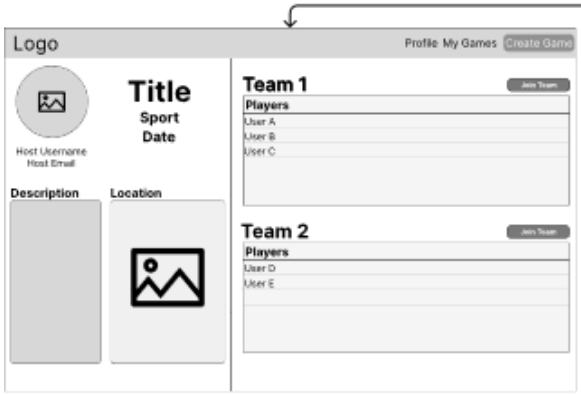

**Mary Clicks Login.**

**Mary Enters her Information and Logs in.**




**Mary Clicks Create Game.**

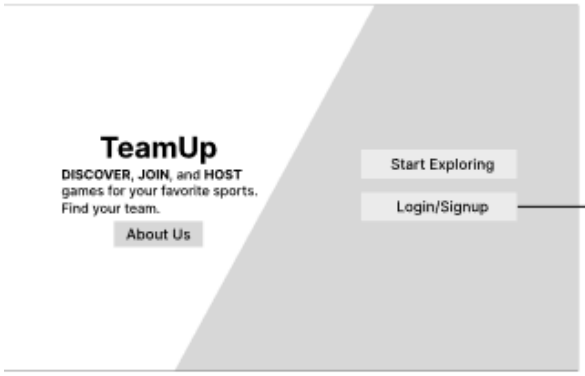
**Mary Enters the required game info and clicks Publish.**



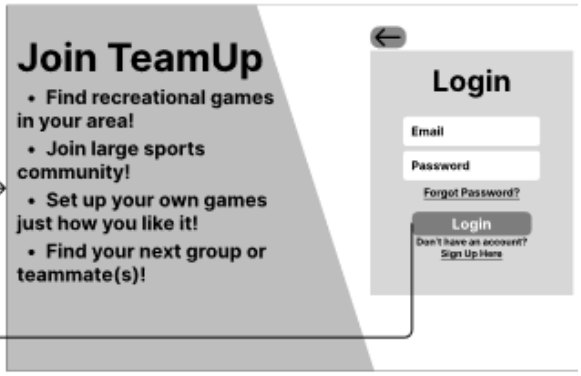

**Mary's game is now created. She waits for an opponent to join.**

## Use Case 5

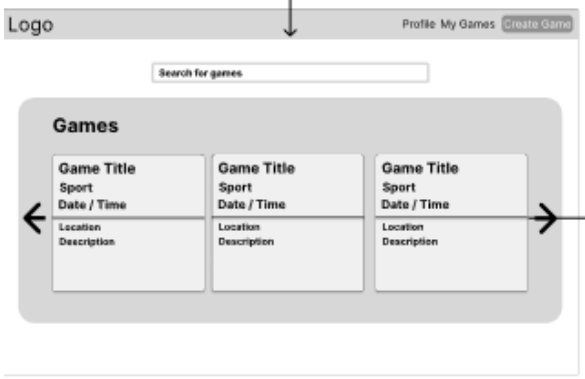
- **Jack has moved to a new city**
- **Jack wants to find a casual basketball match to join and play.**




**Jack Clicks Login.**




**Jack Enters her Information and Logs in.**




**Jack Clicks A Game that Interests him.**

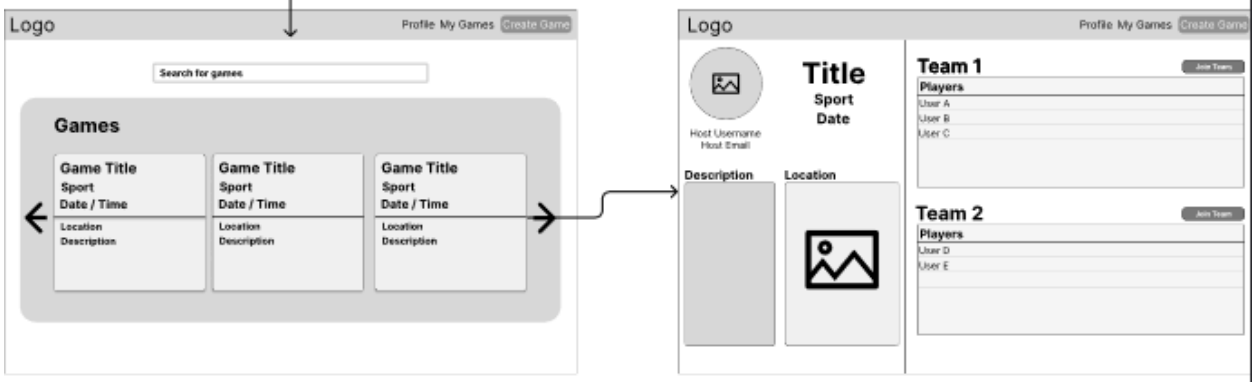

**The description of the game details the skill level. So Jack joins the game.**

## Use Case 6

- Tyler has joined a game using TeamUp
- His roommate, Chris, also wants to play on the same team.


Chris Clicks Login.

Chris Enters her Information and Logs in.

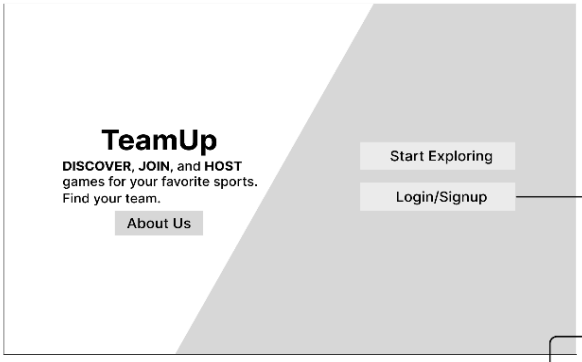



Chris Clicks The Game that Tyler Joined

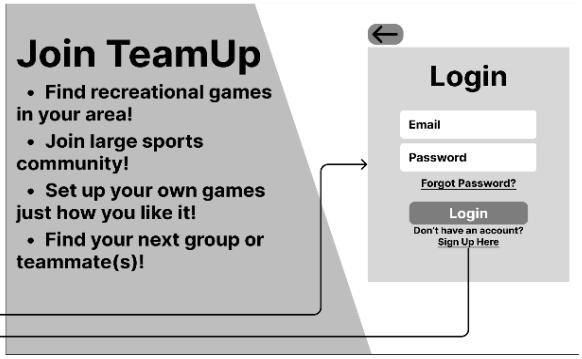
Chris Joins the team that Tyler is on

# Use Case 7

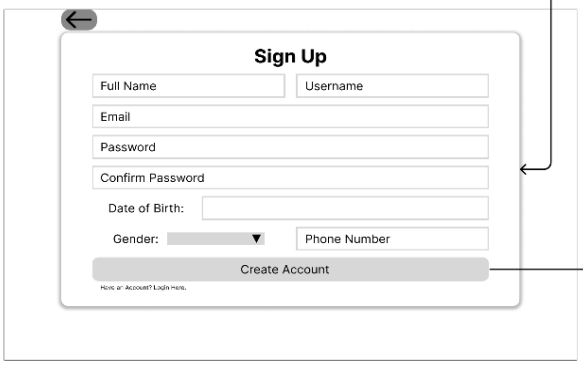
- **Jane moves to San Francisco**
- **Jane plays cricket, but cannot find any games nearby**
- **Jane comes across TeamUp online and creates an account**



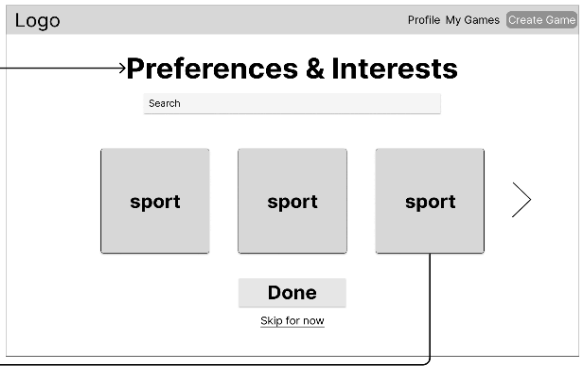

***Jane clicks login/sign up***



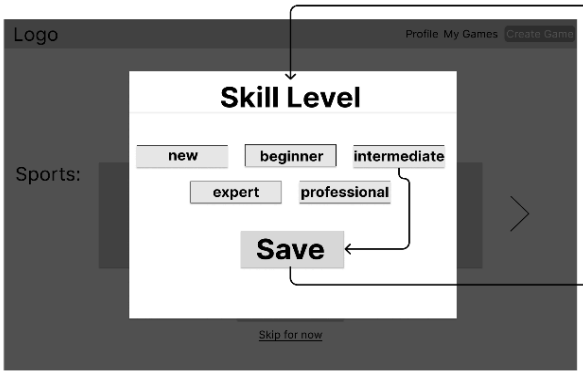

***Jane clicks sign up to create an account***



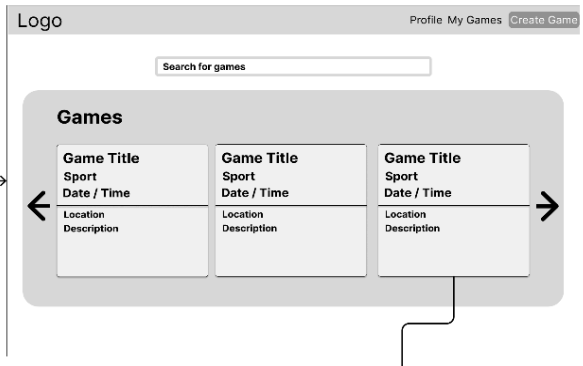

***Jane fills the fields to create an account and is redirected to select her preferences***




***Jane selects her preferred sport and skill level for that sport***

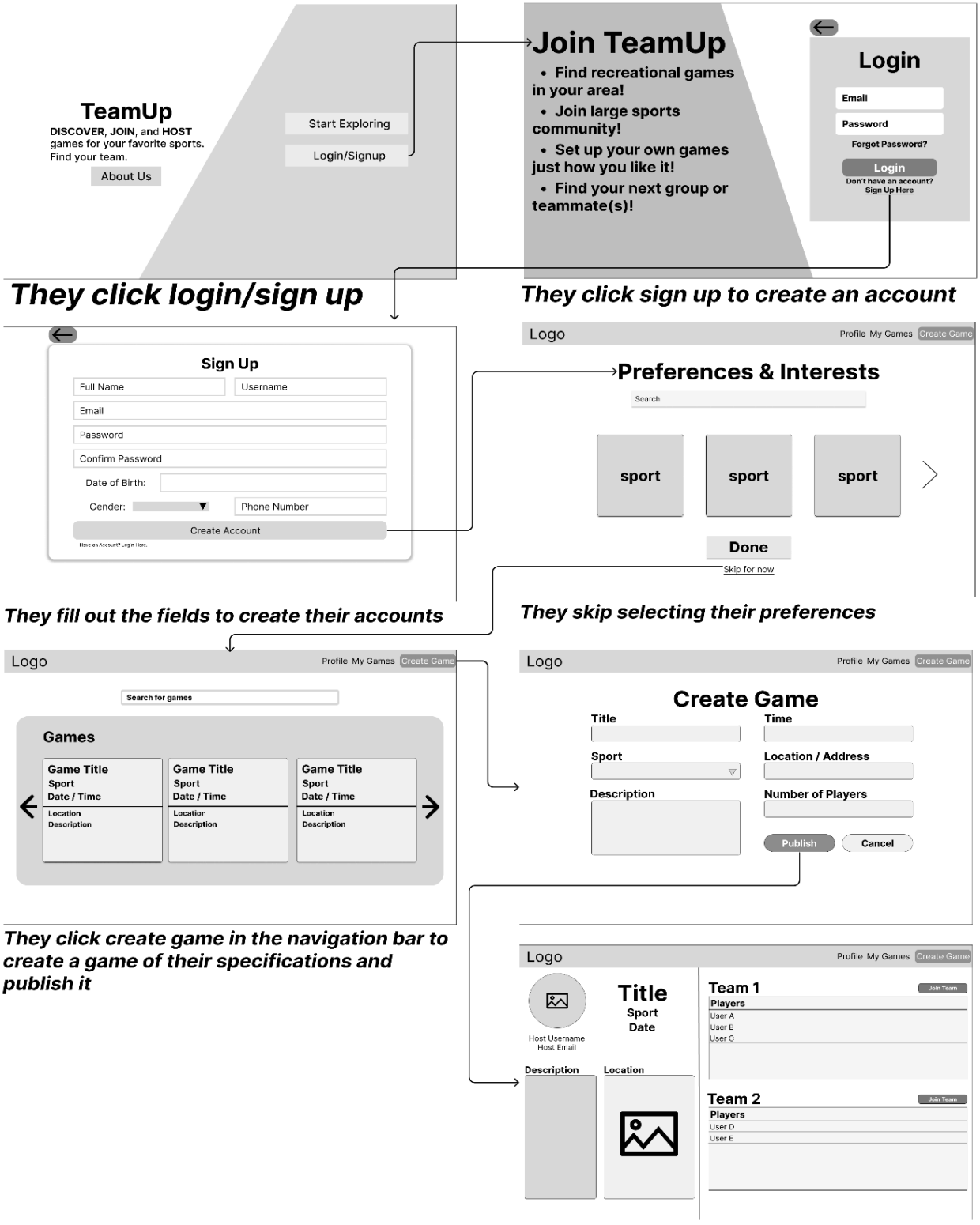



***Jane is redirected to the home page and finds a game to join***




## Use Case 8

- **Trevor and Noah have trouble finding badminton games in their city**
- **They discover TeamUp and find that they can set up and find games that suit their availability**
- **Trevor and Noah go through the same registration process**



# 4. High-level database architecture and organization

**Functional requirements:**

1. **User**
   - 1.1 a user shall be able to join many games.
   - 1.3 a user shall host many games.
   - 1.4 a user shall have many reviews
   - 1.5 a user shall join many teams
   - 1.6 a user shall have many teammates
   - 1.7 a user shall have many sport levels.
   - 1.8 a user shall have many sport preferences.
   - 1.9 a user shall have many tokens.
   - 1.10 a user shall write many reviews.

2. **Sport**
   - 2.1 a sport shall be played by many teams.
   - 2.2 a sport shall be chosen by many games.
   - 2.3 a sport shall be chosen as a preference by many users.

3. **Admin**
   - 3.1 an admin is a user.
   - 3.2 an admin shall manage only one school.
   - 3.3 an admin shall be able to create many tournaments.

4. **Game**
   - 4.1 a game shall be hosted by only one user.
   - 4.2 a game shall have multiple players.
   - 4.3 a game shall specify a type of sport.
   - 4.4 a game shall specify the location.
   - 4.5 a game shall be played by at least one team.

5. **Team**
   - 5.1 a team shall have at least one user.
   - 5.2 a team shall specify a type of sport.
   - 5.3 a team shall play many games.
   - 5.4 a team shall have only one name.

6. **School**

- 6.1 a school shall have only one admin.
- 6.2 a school shall host many tournaments.

7. **Game_Location**
   - 7.1 a game location shall be chosen by many games.
   - 7.2 a game location shall belong to only one city.

8. **Region**
   - 8.1 a region shall have multiple cities.

9. **City**
   - 9.1 a city shall belong to only one region.
   - 9.2 a city shall have multiple game locations.

10. **Review**
    - 10.1 a review shall belong to only one user.
    - 10.2 a review shall be written by only one user.

11. **Token**
    - 11.1 a token shall belong to only one user.

**Non-functional requirements:**

1. **Performance**
   - 1.1 The database shall retrieve data fast.
   - 1.2 The database system shall support concurrent transactions.
   - 1.3 Database queries and server-side operations shall be optimized to minimize latency and ensure smooth user experience.

2. **Storage**
   - 2.1 The database shall support persistent storage
   - 2.2 The database shall store data effectively

3. **Security**
   - 3.1 Data such as password and personal information shall be encrypted
   - 3.2 The access control for personal information shall be strictly managed.
   - 3.3 Data collected shall only be used for making game and tournament management easy and shall not be shared with third parties without user consent.
   - 3.4 Accessing to database shall be limited to specific users such as admin

4. **Scalability**
    - 4.1 The database shall scale easily as the service grows.
    - 4.2 The database shall maintain scalable performance when the accesses increase

5. **Capability**
    - 5.1 The database shall be available all the time
    - 5.2 The database system shall support regular backups of data.

6. **Reliability**
    - 6.1 The system shall be available 99.9% of the time during peak usage hours.
    - 6.2 The application shall be resilient to server failures and able to recover gracefully without data loss.
    - 6.3 Maintenance shall be done during off-peak hours

7. **Usability**
    - 7.1 The user interface shall be easy to use for admins.
    - 7.2 The speed of retrieving data from the database shall be fast for users.

Our group decided to use MySql as our DBMS because it is high performance, high scalability, strong security, and can run on multiple platforms. Also, most people in our group are already familiar with it.

**Entity description:**

1. **User (Strong)**
    - user_id: PK, numeric
    - name: composite (first name, last name), alphanumeric
    - username: alphanumeric
    - email: alphanumeric, email
    - password: alphanumeric
    - dob: multi-value, date
    - Gender: char
    - Phone_number: string
    - created_at: date
    - updated_at: date

- isEmailVerified: boolean
- role: enum

User - Admin is a One-to-One relationship.
User - Review is a Many-to-One relationship.
User - Sport is a Many-to-Many relationship.
User - Game is a Many-to-One relationship.
User - Team is a Many-to-Many relationship.

2. **Sport (Strong)**
    - sport_id: PK, numeric
    - sport_name: alphanumeric
    - description: alphanumeric

Sport - User is a Many-to-Many relationship.
Sport - Game is a Many-to-One relationship.
Sport - Team is a Many-to-One relationship.

3. **Admin (Weak)**
    - admin_id: PK, numeric
    - user_id: FK, numeric
    - school_id : FK, numeric

Admin - School is a One-to-One relationship.
Admin - User is a One-to-One relationship.

4. **Game (Weak)**
    - game_id: PK, numeric
    - sport_id: FK, numeric
    - date_time: date
    - number_of_players: Numeric
    - game_location_id: FK, numeric
    - description: alphanumeric
    - fee: alphanumeric
    - gender: alphanumeric
    - age_group: numeric
    - user_id: FK, numeric
    - team_id: FK, numeric

Game - User is a One-to-Many relationship.

Game - Sport is a One-to-Many relationship.
Game - Game Location is a One-to-Many relationship.
Game - Team is a Many-to-Many relationship.

5. **Team (Weak)**
    - team_id: PK, numeric
    - name: alphanumeric
    - sport_id: FK, numeric
    - user_id: FK, numeric

Team - User is a Many-to-Many relationship.
Team - Sport is a One-to-Many relationship.
Team - Game is a Many-to-Many relationship.

6. **School (Strong)**
    - school_id: PK, numeric
    - name: alphanumeric
    - location: alphanumeric

School - Admin is a One-to-One relationship.

7. **Game_Location (Weak)**
    - location_id: PK, numeric
    - name: alphanumeric
    - address: alphanumeric
    - description: alphanumeric
    - parking: alphanumeric
    - fee: alphanumeric
    - map_url: alphanumeric
    - city_id: FK, numeric

Game Location - City is a One-to-Many relationship.
Game Location - Game is a Many-to-One relationship.

8. **Region (Strong)**
    - region_id: PK, numeric

Region - City is a Many-to-One relationship.

9. **City (Weak)**

- region_id: FK, numeric
- city_id: PK, numeric

City - Region is a One-to-Many relationship.
City - Game Location is a Many-to-One relationship.

**10. Review (Weak)**

- review_id: PK, numeric
- user_id: FK, numeric
- rating: alphanumeric
- description: alphanumeric

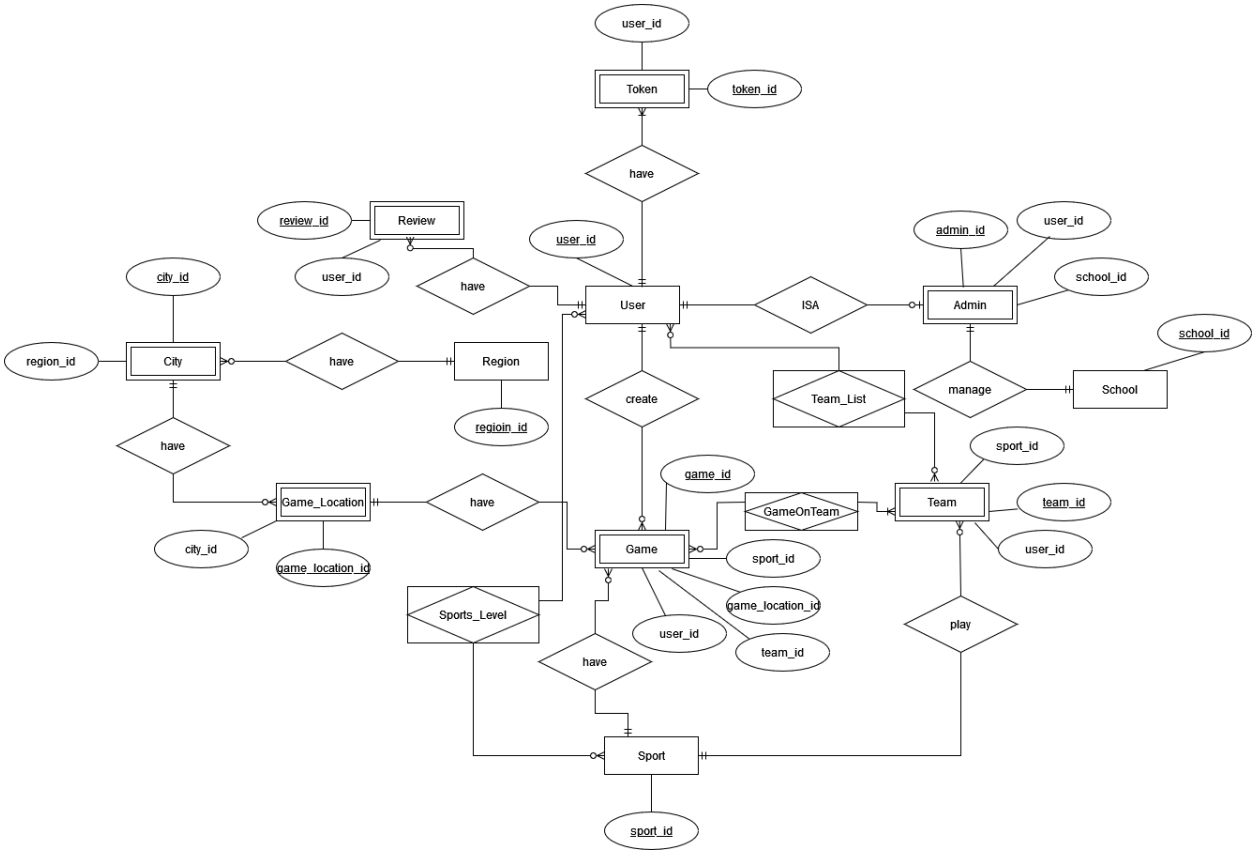
Review - User is a One-to-Many relationship.

**11. Token (Weak)**

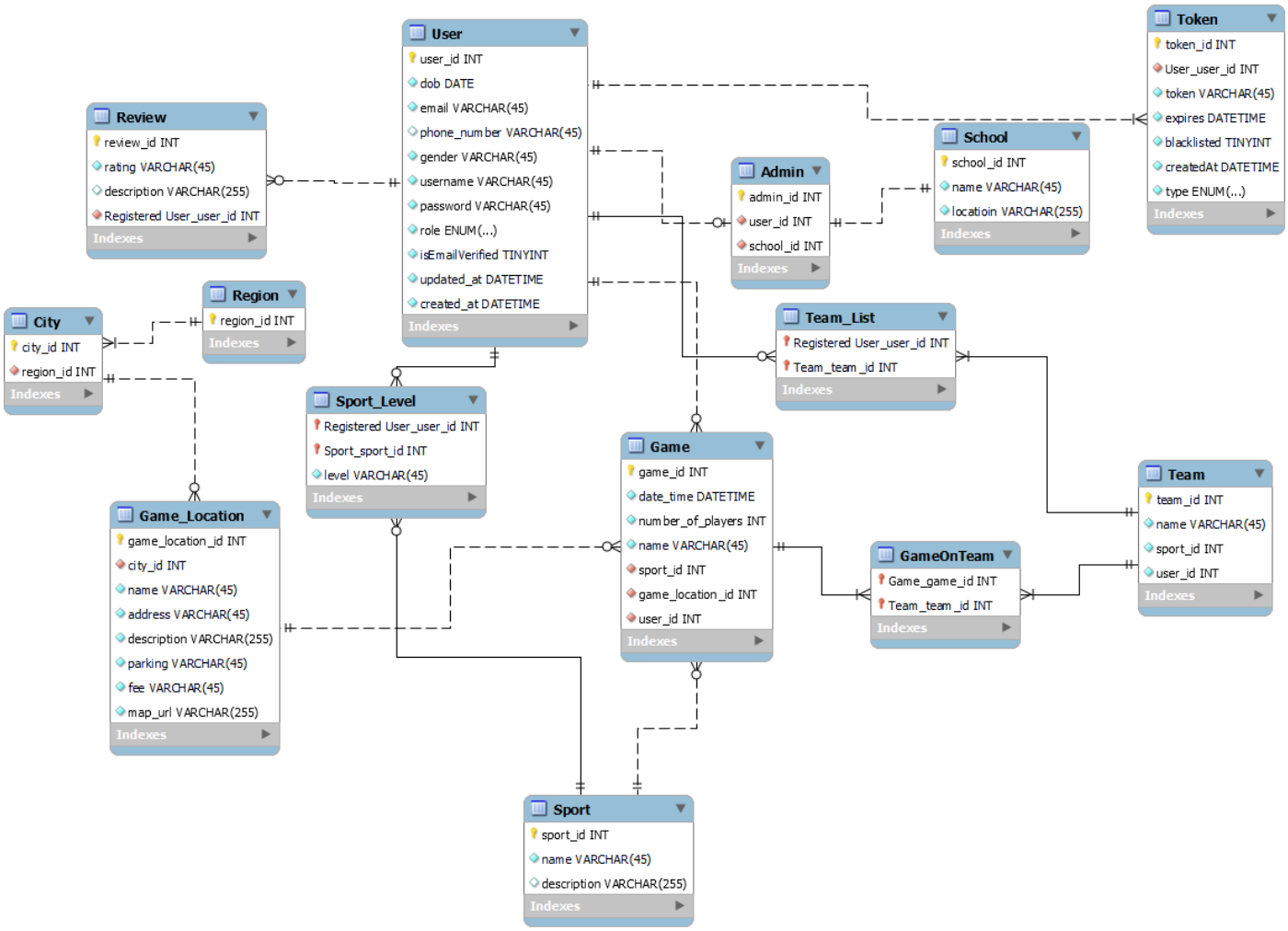
- token_id: PK
- user_id: FK
- token: alphanumeric
- type: enum
- expires: date
- blacklisted: boolean
- createdAt: date

Token - User is a One-to-Many relationship.

**ERD:**

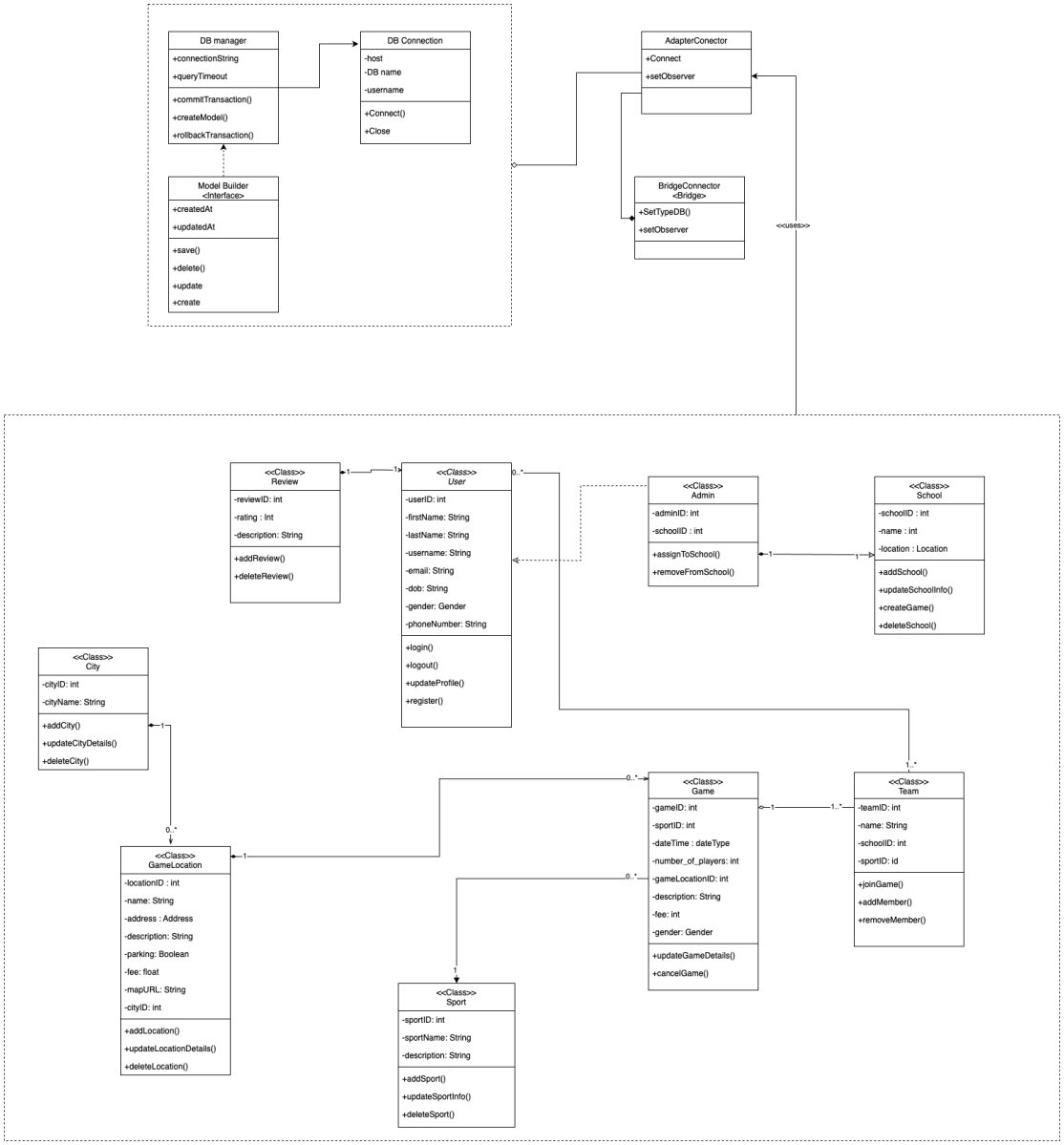

user_id
Token
token_id
have
review_id
Review
user_id
have
user_id
User
ISA
admin_id
user_id
Admin
school_id
city_id
region_id
City
have
Region
regioin_id
have
create
Team_List
manage
School
school_id
sport_id
game_id
Game_Location
have
GameOnTeam
Team
team_id
city_id
game_location_id
Game
sport_id
user_id
Sports_Level
game_location_id
user_id
team_id
play
have
Sport
sport_id

**EER:**

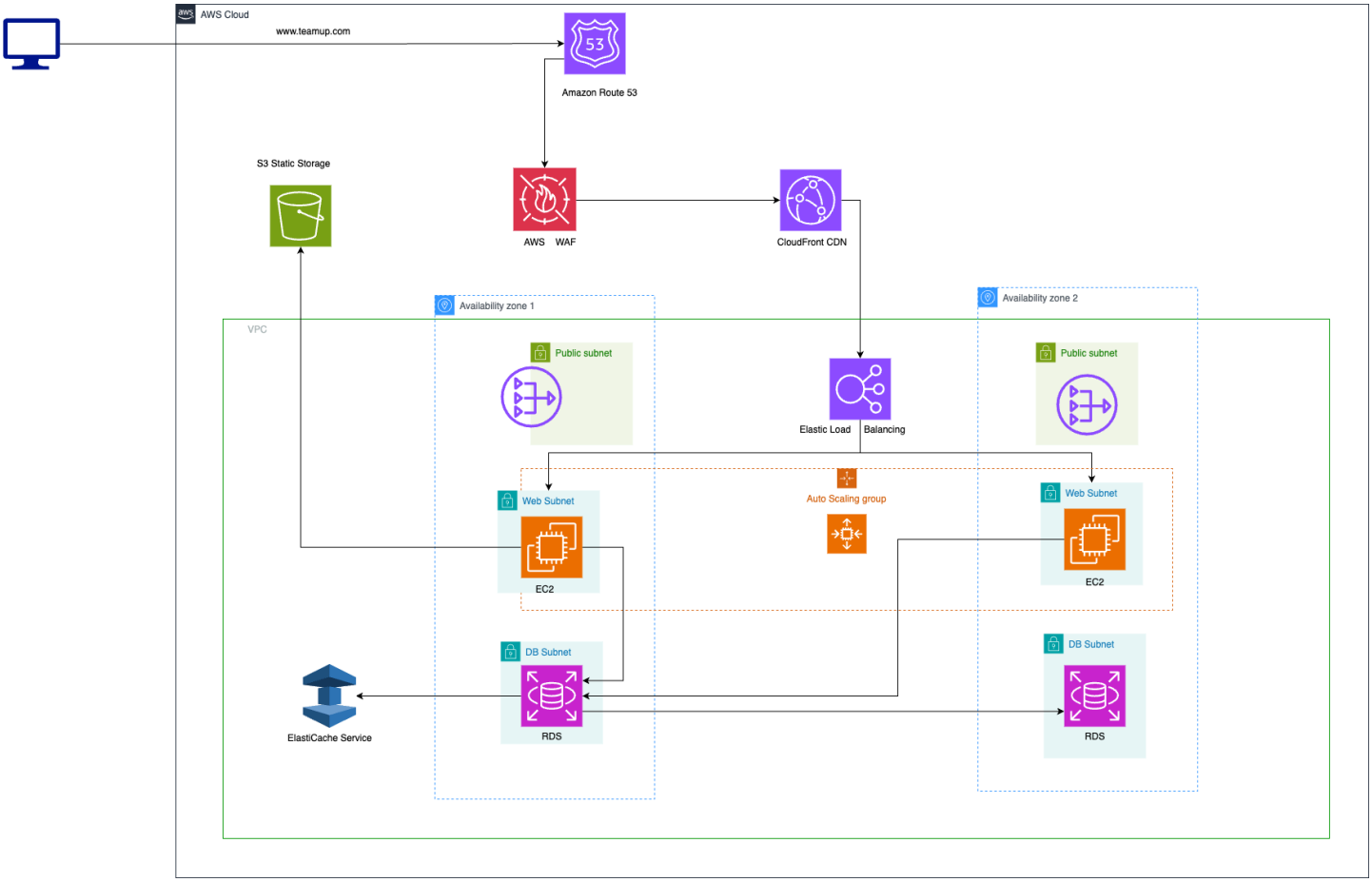

User
user_id INT
dob DATE
email VARCHAR(45)
phone_number VARCHAR(45)
gender VARCHAR(45)
username VARCHAR(45)
password VARCHAR(45)
role ENUM(...)
isEmailVerified TINYINT
updated_at DATETIME
created_at DATETIME
Indexes
Token
token_id INT
User_user_id INT
token VARCHAR(45)
expires DATETIME
blacklisted TINYINT
createdAt DATETIME
type ENUM(...)
Indexes
Review
review_id INT
rating VARCHAR(45)
description VARCHAR(255)
Registered User_user_id INT
Indexes
School
school_id INT
name VARCHAR(45)
locatioin VARCHAR(255)
Indexes
Admin
admin_id INT
user_id INT
school_id INT
Indexes
Region
region_id INT
Indexes
City
city_id INT
region_id INT
Indexes
Team_List
Registered User_user_id INT
Team_team_id INT
Indexes
Sport_Level
Registered User_user_id INT
Sport_sport_id INT
level VARCHAR(45)
Indexes
Game
game_id INT
date_time DATETIME
number_of_players INT
name VARCHAR(45)
sport_id INT
game_location_id INT
user_id INT
Indexes
Team
team_id INT
name VARCHAR(45)
sport_id INT
user_id INT
Indexes
Game_Location
game_location_id INT
city_id INT
name VARCHAR(45)
address VARCHAR(45)
description VARCHAR(255)
parking VARCHAR(45)
fee VARCHAR(45)
map_url VARCHAR(255)
Indexes
GameOnTeam
Game_game_id INT
Team_team_id INT
Indexes
Sport
sport_id INT
name VARCHAR(45)
description VARCHAR(255)
Indexes

# 5. High-Level Diagrams

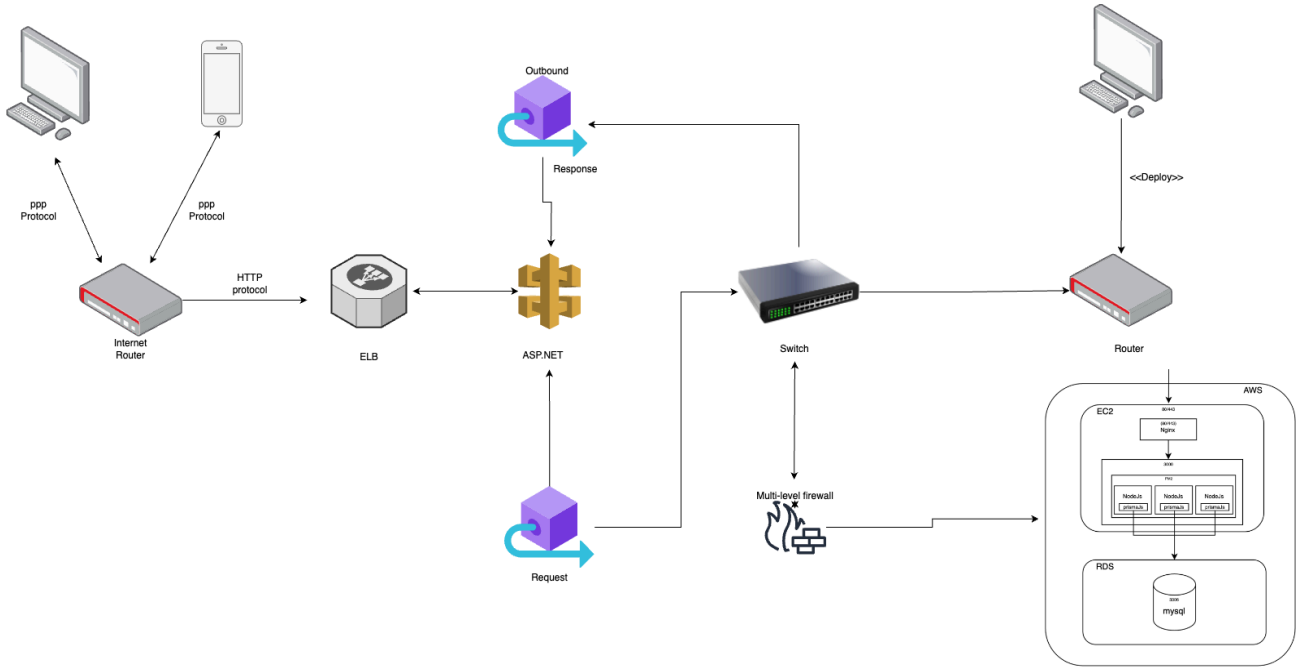
## UML Class diagrams



## Scalability Diagram



Application Network and Deployment Design:



# 6. Detailed list of contributions

| | | |
|---|---|---|
| Juan Estrada | 9 | • Data Definitions V2<br>• Application Network and Deployment Design.<br>• [API] Game Listing Details.<br>• [API] Get Joined Games |
| Kotaro Iwanaga | 9 | • [API] Get Joined Games<br>• [API]  Get, update, and delete user's preferences<br>• [API] for getting games user-hosted. |
| Cole Chiodo | 9 | • ⅓ Figma wireframes<br>• My profile page<br>• My games page |
| Martin Pham | 9 | • ⅓Figma wireframes<br>• Signup page v2<br>• Preferences page<br>• Editing Preferences page<br>• Migration of existing pages to nextjs |
| Jaycee Lorenzo | 9 | • ⅓Figma wireframes<br>• Detail game page<br>• Create game page<br>• Card component<br>• Landing page v2 |
| Areeb abbasi | 8 | • Automatic deployment |

**Prototype In-Class Review Notes**

- Title Page
  - 
- Main
  - Search suggestions even if no results
- Detailed
  - “Dont know if screen problems or bad design”
  - What if someone random joins your team when you dont want them?
    - Host privileges?
- Login
  - 
- Signup
  - Username sticking out
  - Link to tos policy
  - Error show after clicking login
- Create Game
  - Negative number of players
- My Games
  - 
- Profile
  - 
- Interests
  - 
- Other
  - Social Media Links
  - Fix for all screen sizes
  - Footer links to wrong things
    - Survey link
  - Slow loading
    - > 1 sec bad
  - UI Good, UX Problems

SW Engineering CSC648-848-05 Spring 2024

Application Title & Name : Teamup

Team : 5

| Student Name | SFSU Email | GitHub | Discord | Role |
|---|---|---|---|---|
| Juan Estrada | jestradazuluaga@sfsu.edu | jjestrada2 | juanjosee | Team-lead |
| Areeb Abbasi | aabbasi@sfsu.edu | areeeeb | _xertz | Backend-lead |
| Cole Chiodo | cchiodo@sfsu.edu | colechiodo | colechiodo | Docs-editor |
| Jaycee Lorenzo | jlorenzo3@sfsu.edu | jclorenz0 | __jaycee | Frontend-lead |
| Martin Pham | mpham8@sfsu.edu | mar10fam | marnoki | Github-master |
| Kotaro Iwanaga | kiwanaga@sfsu.edu | iamkotaaa | kotaro8448 | Database-admin |

Milestone 4

| Version # | Submission Date |
|---|---|
| M4V2 | 05/22/2024 |
| M4V1 | 05/16/2024 |
| M3V2 | 05/15/2024 |
| M3V1 | 04/25/2024 |
| M2V2 | 04/18/2024 |
| M2V1 | 04/04/2024 |
| M1V2 | 03/21/2024 |
| M1V1 | 03/01/2024 |

# Content Table

# 1) Product Summary

- **Product:** TeamUp

- **Major Functions:**

### User:

1. Users shall be able to search for games based on sport type.
2. Users shall be able to search for games based on location.
3. Users shall be able to search for games based on time.
4. Users shall be able to join a team in a game.
5. Users shall be able to create new games.
6. Users shall be able to specify the sport for a game.
7. Users shall be able to specify the location of a game.
8. Users shall be able to specify the time of a game.
9. Users shall be able to specify the number of players needed for a game.
10. Users shall be allowed to update their account profile picture.
11. Users shall be allowed to update their account biography description.
12. Users shall have the option to manually log out from their accounts.
13. Users shall have the option to verify their accounts through email.
14. Users shall be able to recover their password through email.
15. Users shall be able to detach from a game.
16. Users shall be able to search all their joined games.
17. Users shall be able to search all their hosted games.

### Game listings:

1. Game listings shall be able to provide a facility location map.
2. Game listings shall be able to provide the player's username.
3. Game listings shall be able to provide player’s pictures.
4. Game listings shall be able to provide the player’s link to their biography.
5. Game listings shall be able to provide facility game rules.
6. Game listings shall be able to provide the name of the organizer.
7. Game listings shall be able to provide the name of the game.
8. Game listings shall be able to provide the day and time of the game.
9. Game listings shall be able to provide the contact information of the organizer.
10. Game listings shall be able to provide a short description of the game.
11. Game listings shall be able to provide the empty slots of the game.

### Game Location:

1. Game locations shall be able to show address and facility details.
2. Game locations shall be able to provide parking information.
3. Game locations shall be able to provide users with reviews.

### Sport:

1. Users shall be able to choose sports preferences
2. Users shall be able to select their skill level in a specific sport.

### Team:

1. Teams shall be able to be formed in a game listing.
2. Teams shall be restricted by a specific number of players in a game.

### Profile:

1. Profile preferences shall be able to be edited by the user.
2. Profile pictures shall be able to be uploaded to the user's bio(JPG, 320x320px).
3. Profile email shall be able to be edited by the user.
4. Profile phone number shall be able to be edited by the user.
5. Profile gender shall be able to be edited by the user.
6. Profile birthday shall be able to be edited by the user.
7. Profile password shall be able to be edited by the user.

### Review:

1. Users shall be able to rate other players.
2. Users shall be able to write reviews for other players.
3. Users shall be able to rate facilities for game locations.
4. Users shall be able to write reviews for game locations.

## • Unique Features:

The unique feature of our app is the ability to use the app to find and join casual sports matches. Our competitors, such as SportsEngine and ZogSports are only for sport tournaments. Our app allows the ability to create and join casual, non-competitive sport matches. Other competitors, like Plei, only focus on a single sport, Soccer. This does not cover casual matches for other sports. And websites like Meetup allow any type of meetup, not just sports. Our app focuses just on sports so it has a better experience for sports.

Another unique feature is the ability to review game locations and users. You can review game locations on aspects such as facility quality or nearby amenities. You can also review other users based on their skill level or attitude. Some of our competitors do offer the ability to give accolades to other users, but not to give a star rating and written review.

- **Product URL:** http://44.220.154.73/

# 2) Usability test plan

## **Test 1:** Search Bar for Games

**What is being tested:** The search bar's functionality, accuracy, and ease of use.

**Why:** The search bar is a critical feature that differentiates the app from competitors. It's likely the first function users will interact with, and its performance impacts their initial impression. Additionally, it employs a custom algorithm to match desired games, which needs to be validated for effectiveness in a real-world setting with substantial data.

**System Setup:**
- The test will be conducted on a personal computer.
- The application URL will be accessed via Google Chrome.

**Starting Point:**
- Open a new tab in Google Chrome.
- Enter the application URL in the search bar and navigate to the homepage.

**Intended Users:**
- User 1 (Ivy)
- User 2 (Ishan)
- User 3 (Ralph)
- User 4 (Maynard)

**URL :** http://44.220.154.73/

**Objective:** To evaluate the usability of the search bar for locating games nearby the user's location.

**Test Description:** The user must be able to use the search bar function to find a certain game

**What is to be Measured:**
- Effectiveness: Accuracy of search results in finding relevant games nearby.
- Efficiency: Time taken to locate and select a game from the search results.
- User Satisfaction: Users' subjective satisfaction with the search bar's performance and usability.

| User | Task Completion (Y/N) | Errors made | Accuracy (1-5) | Time (Sec.) | Steps to Completion | Effectiveness (1-5) | Efficiency | Satisfaction |
|---|---|---|---|---|---|---|---|---|
| Ivy | Y | 0 | 5 | 10.86s | 3 | 5 | 5 | 4 |
| Ralph | Y | 0 | 5 | 15.45s | 3 | 5 | 5 | 4 |
| Ishan | Y | 0 | 5 | 9.20s | 3 | 5 | 5 | 3 |
| Maynard | Y | 0 | 5 | 8.69s | 3 | 5 | 5 | 4 |

## Feedback

**Likert Scale Questionnaire:**

- The search bar was easy to use.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Strongly Agree |

- The search results were accurate and relevant.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Strongly Agree |

- I am satisfied with the overall performance of the search bar.

Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Agree |

**Errors users had:**

- No result popped up when searching ‘pickleball’ (See below for comments).

**Comments:**

- Developer Comments:
    - Search results solely depend on the user base as they are the source of the creation of games.
    - Also depends on supported sports by our application. In this test case, users were searching for pickleball games but could not find any.
        - Reason 1: No pickleball games posted.
        - Reason 2: Pickleball, as a sport selection, is not supported yet.

---

## **Test 2:** Joining a game

**What is being tested:** The functionality of the "Join Game" button.

**Why:** This feature is crucial because after users find a game they are interested in, the ability to join it is the next critical step. If this functionality is broken or difficult to use, users are likely to leave the app, which would go against the app's promise on the landing page: "DISCOVER, JOIN, and HOST games for your favorite sports."

**System Setup:**
- The test will be conducted on a personal computer.
- The application URL will be accessed via Google Chrome.

**Starting Point:**
- Open a new tab in Google Chrome.
- Enter the application URL in the search bar and navigate to the homepage.
- Perform a search to find a game using the search bar and select a specific game.

**Intended Users:**
- User 1 (Ivy)
- User 2 (Ishan)
- User 3 (Ralph)
- User 4 (Maynard)

**URL :** http://44.220.154.73/home

**Objective:** To evaluate the usability of the button to join a team in a specific game.

**Test Description:** The user must be able to join a specific team in a game in an intuitive way.

**What is to be Measured:**

- Effectiveness: Success rate of users joining a game using the button.
- Efficiency: Time taken to complete the process of joining a game.
- User Satisfaction: Users' subjective satisfaction with the ease of joining a game.

| User | Task Completion (Y/N) | Errors made | Accuracy (1-5) | Time (Sec.) | Steps to Completion | Effectiveness (1-5) | Efficiency | Satisfaction |
|---|---|---|---|---|---|---|---|---|
| Ivy | Y | 0 | 4 | 12.5s | 2 | 5 | 4 | 4 |
| Ralph | Y | 0 | 5 | 5.06s | 2 | 5 | 3 | 3 |
| Ishan | Y | 0 | 5 | 11.05s | 4 | 5 | 4 | 4 |
| Maynard | Y | 0 | 5 | 3.21s | 2 | 5 | 5 | 5 |

## Feedback

**Likert Scale Questionnaire:**

- The process of joining a game was easy and straightforward
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Strongly Agree |

- I am satisfied with the overall functionality of the "Join Game" button.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Strongly Agree |

- I was able to easily find, view, and manage the games that I have joined.

| User | Answer |
|---|---|

| | |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Agree |
| Ishan | Strongly Agree |
| Maynard | Strongly Agree |

**Errors users had:**

- N/A

**Comments:**

- Ivy's Comments:
    - 'Join Game' button was not very obvious.

---

# **Test 3:** Create Game Form

**What is being tested:** The functionality and user experience of the form used to create a new game.

**Why:** This function is crucial because, to maintain user engagement, we need users to take the initiative to create games. Ensuring that this process is smooth and user-friendly will help increase the number of games on the platform, thereby enhancing the app's popularity and competitive edge in the market.

**System Setup:**

- The test will be conducted on a personal computer.
- The application URL will be accessed via Google Chrome.

**Starting Point:**

- Open a new tab in Google Chrome.
- Enter the application URL in the search bar and navigate to the homepage.
- Navigate to the "Create Game" page from the homepage.

**Intended Users:**

- User 1 (Ivy)
- User 2 (Ishan)
- User 3 (Ralph)
- User 4 (Maynard)

**URL :** http://44.220.154.73/home

**Objective:** To evaluate the usability of the "Create Game" form.

**Test Description:** The user must be able to input title , sport, description date, time, location and number of players and then be redirected to the page in which they can look at their joined and hosted games.

**What is to be Measured:**

- Effectiveness: Success rate of users completing the form and creating a game.
- Efficiency: Time taken to complete the form and create a game.
- User Satisfaction: Users' subjective satisfaction with the ease of creating a game and being redirected to the page where they can view their joined and hosted games.

| User | Task Completion (Y/N) | Errors made | Accuracy (1-5) | Time (Sec.) | Steps to Completion | Effectiveness (1-5) | Efficiency | Satisfaction |
|---|---|---|---|---|---|---|---|---|
| Ivy | Y | 0 | 5 | 46.20s | 9 | 5 | 4 | 5 |
| Ralph | Y | 1 | 4 | 44.93s | 10 | 4 | 4 | 4 |
| Ishan | Y | 0 | 5 | 41.06s | 9 | 5 | 4 | 4 |
| Maynard | Y | 3 | 4 | 1:33.75s | 11 | 3 | 3 | 3 |

## Feedback

**Likert Scale Questionnaire:**

- The form to create a game was easy to complete.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Neutral |

- The information fields in the form were clear and easy to understand.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Agree |
| Ishan | Strongly Agree |
| Maynard | Neutral |

- I am satisfied with the overall experience of creating a game and being redirected to the detailed game page.

Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Agree |
| Ishan | Strongly Agree |
| Maynard | Agree |

**Errors users had:**

- No results for the desired sport, pickleball, available when choosing sport for the game.
- Not all location results were showing when typing in desired locations.
- Time component was difficult to use.

**Comments:**

- Ivy's Comments:
  - Make it more obvious that sports and locations can be searched by changing labels.
- Ralph's Comments:
  - Need more sports and locations for games.
  - Make it more obvious that the location dropdown is scrollable.
- Ishan's Comments:
  - Change time selection/input component to make it more usable.
- Developer Comments:
  - Users had some trouble using the time component.
  - Need to support more locations.
  - Possibly implement a way for users to add a location if the desired location is not present.

---

# **Test 4:** Give a review to a user

**What is being tested:** The functionality that allows users to give a review and rate other users.

**Why:** This function is important because it fosters interaction and engagement on the platform, which can lead to increased user retention and growth. Allowing users to give reviews and rate others enhances the app's credibility and creates a sense of active, genuine user participation, rather than relying on fake data to simulate activity.

**System Setup:**

- The test will be conducted on a personal computer.
- The application URL will be accessed via Google Chrome.

**Starting Point:**

- Open a new tab in Google Chrome.
- Enter the application URL in the search bar and navigate to the homepage.
- Navigate to a Game page from the homepage.
- Navigate to a user's profile page where the review and rating can be submitted.

**Intended Users:**

- User 1 (Ivy)
- User 2 (Ishan)
- User 3 (Ralph)
- User 4 (Maynard)

**URL :** http://44.220.154.73/home

**Objective:** To evaluate the usability of the review and rating feature for users.

**Test Description:** The user must be able to input the number of stars and a short comment and post the comment to the user's profile.

**What is to be Measured:**

- Effectiveness: Success rate of users submitting a review and rating.
- Efficiency: Time taken to complete and submit the review and rating.
- User Satisfaction: Users' subjective satisfaction with the ease of submitting a review and rating.

| User | Task Completion (Y/N) | Errors made | Accuracy (1-5) | Time (Sec.) | Steps to Completion | Effectiveness (1-5) | Efficiency | Satisfaction |
|---|---|---|---|---|---|---|---|---|
| Ivy | Y | 0 | 5 | 32.75s | 4 | 5 | 4 | 5 |
| Ralph | Y | 1 | 4 | 56.43s | 5 | 5 | 4 | 4 |
| Ishan | Y | 0 | 5 | 43.05s | 4 | 5 | 4 | 4 |
| Maynard | Y | 0 | 5 | 51.86s | 4 | 5 | 4 | 4 |

## Feedback

**Likert Scale Questionnaire:**

- The process of giving a review and rating another user was easy and straightforward. Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Agree |
| Ralph | Strongly Agree |
| Ishan | Agree |
| Maynard | Agree |

- The review and rating feature was clearly visible and accessible on the user profile page. Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Agree |
| Ishan | Agree |
| Maynard | Agree |

- I am satisfied with the overall experience of giving a review and rating another user.

Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Agree |
| Maynard | Strongly Agree |

**Errors users had:**

- Did not select the number of stars before submitting the review.

**Comments:**

- Ralph's Comments:
  - Make the 'stars' component of creating a review more visible/obvious. It is hard to see because of the color combination of the stars and background.
- Ishan's Comments:
  - Star rating component was not very obvious. Realized/noticed it while writing the review description.

---

# **Test 5:** Users shall be able to detach from a game.

**What is being tested:** The functionality that allows users to give a review and rate game location.
**Why:** This function is important because it can facilitate future partnerships with businesses and provide a potential revenue stream by recommending certain game locations. User-generated reviews also enhance the credibility and usefulness of the app, encouraging more users to rely on it for finding suitable game locations.

**System Setup:**
- The test will be conducted on a personal computer.
- The application URL will be accessed via Google Chrome.

**Starting Point:**
- Open a new tab in Google Chrome.
- Enter the application URL in the search bar and navigate to the homepage.
- Navigate to a Game page from the homepage.
- Navigate to a game location profile page where the review and rating can be submitted.

**Intended Users:**
- User 1 (Ivy)
- User 2 (Ishan)
- User 3 (Ralph)
- User 4 (Maynard)

**URL :** http://44.220.154.73/home

**Objective:** To evaluate the usability of the review and rating feature for game locations.

**Test Description:** The user must be able to input number of stars and a short comment and post the commend to game location.

**What is to be Measured:**
- Effectiveness: Success rate of users submitting a review and rating for game locations.
- Efficiency: Time taken to complete and submit the review and rating for game locations.
- User Satisfaction: Users' subjective satisfaction with the ease of submitting a review and rating for game locations.

| User | Task Completion (Y/N) | Errors made | Accuracy (1-5) | Time (Sec.) | Steps to Completion | Effectiveness (1-5) | Efficiency | Satisfaction |
|---|---|---|---|---|---|---|---|---|
| Ivy | Y | 0 | 5 | 8.55s | 3 | 5 | 4 | 5 |
| Ralph | Y | 0 | 5 | 10.03s | 3 | 5 | 4 | 5 |
| Ishan | Y | 0 | 5 | 9.92s | 3 | 5 | 4 | 4 |
| Maynard | N | 1 | - | 12.43s | - | 1 | 4 | 3 |

## Feedback

**Likert Scale Questionnaire:**

- The process of leaving a joined game was easy and straightforward.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | Disagree |

- I am satisfied with the location and visibility of the ‘Leave Game’ button.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Strongly Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | - |

- The option to leave a joined game was made obvious to me.
  Strongly Agree / Agree / Neutral / Disagree / Strongly Disagree

| User | Answer |
|---|---|
| Ivy | Agree |
| Ralph | Strongly Agree |
| Ishan | Strongly Agree |
| Maynard | - |

**Errors users had:**

- Critical error found:
    - When attempting to visit a game that the user has created/hosted, the server would crash.

**Comments:**

- Developer Comments:
    - Fix API that retrieves the games that the user has created/hosted.

# 3) QA test plan

**Test objectives:** To verify the system responds to user interactions within 2 seconds for browsing games.

**HW and SW setup:**

HW setup:
CPU speed: 2.42 HZ
Memory RAM capacity: 16 GB
Hard Drive Storage: 475 GB
The size of the screen: 14 inches

SWsetup:
Operating System: Windows 11 Pro
Browser: Chrome, Firefox, and Edge

Page: Home

Page:edit-preferences

Page:create-game

**Feature to be tested:** Response Time
**QA test plan:** please see the table below

| Test # | Test Title | Test Description | Test Input | Expected Correct Output | Test Results (PASS/FAIL) |
|---|---|---|---|---|---|
| 1 | Clicking on a game in a home page | Click on a panel of a game, and a page of details need to appears within 2 seconds | Click on a game. It can be any game. | Page appears within 2 seconds | Edge:PASS<br>Chrome: PASS<br>Firefox: PASS |
| 2 | Choosing preferences | Edit your preferences and submit your change. It should submit the data within 2 seconds. | Edit your preference and click on Done. | It should submit the change within 2 seconds. | Edge: PASS<br>Chrome: PASS<br>Firefox: PASS |
| 3 | Create a new game | Create a new game, and click on create game. It should make a new game within 2 seconds. | Go to the create-game page and create a new game. | It should make a game within 2 seconds. | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |

**Test objectives:** To verify that responsible design works properly in each browser
**HW and SW setup:**

HW setup:
CPU speed: 2.42 HZ
Memory RAM capacity: 16 GB
Hard Drive Storage: 475 GB
The size of the screen: 14 inches

SW setup:
Operating System: Windows 11 Pro
Browser: Chrome, Firefox, and Edge

Page: login

Page:signup

Page:create-game

**Feature to be tested:** Responsive design of the website (Compatibility)
**QA test plan:** please see the table below

| Test # | Test Title | Test Description | Test Input | Expected Correct Output | Test Results (PASS/FAIL) |
|---|---|---|---|---|---|
| 1 | Testing the responsible design of the signup page | In each browser, set the screen size to the given test input. | Width: 375 pixels<br><br>Height: 812 pixels | All input fields need to be in the form. | Edge: PASS<br>Chrome: PASS<br>Firefox: PASS |
| 2 | Testing the responsible design of the login page | In each browser, set the screen size to the given test input. | Width: 768 pixels<br><br>Height: 1024 pixels | All input fields need to be in the form. | Edge: PASS<br>Chrome: PASS<br>Firefox: PASS |
| 3 | Testing the responsible design of the create-game page | In each browser, set the screen size to the given test input. | Width: 375 pixels<br><br>Height: 812 pixels | All input fields need to be in the form. | Edge: PASS<br>Chrome: PASS<br>Firefox: PASS |

**Test objectives:** To verify the system is available 99.9% of the time during peak usage hours.

**HW and SW setup:**

HW setup:
CPU speed: 2.42 HZ
Memory RAM capacity: 16 GB
Hard Drive Storage: 475 GB
The size of the screen: 14 inches

SW setup:
Operating System: Windows 11 Pro
Browser: Chrome, Firefox, and Edge

Page:
URL:

**Feature to be tested:** Reliability
**QA test plan:** please see the table below

| Test # | Test Title | Test Description | Test Input | Expected Correct Output | Test Results (PASS/FAIL) |
|---|---|---|---|---|---|
| 1 | Clicking on the link of the website with other people | Click on the link of the website with other people. It should take you to the home page without any problems. | Click on the URL | Taking users to home page | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |
| 2 | Making many games | Making many games with other users to test it doesn't break the system | Try to make many games with other users at the same time | It should make games without problems | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |
| 3 | Joining a team and detach many times | Do cycles of joining a team and detaching from that team to make sure it doesn't break the system | Find a game you want to join, and try to join and detach many times with other users | Users should be able to join and leave the team without any problems. Usernames should be shown under teams. | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |

**Test objectives:** To verify that user interactions and workflows shall be consistent across different sections of the application.

**HW and SW setup:**

HW setup:
CPU speed: 2.42 HZ
Memory RAM capacity: 16 GB
Hard Drive Storage: 475 GB
The size of the screen: 14 inches

SW setup:
Operating System: Windows 11 Pro
Browser: Chrome, Firefox, and Edge

Page: create-game

**Feature to be tested:** Usability Requirements
**QA test plan:** Please see the table below

| Test # | Test Title | Test Description | Test Input | Expected Correct Output | Test Results (PASS/FAIL) |
|---|---|---|---|---|---|
| 1 | Invalid date for game creation | When you create a game, try to set a invalid date. It should give an error message. | Invalid date | Enter a valid date | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |
| 2 | Enter a invalid number of players for game creation | When you create a game, try to enter a invalid number of players such as minus | Minus number | Enter a valid number of players | Edge:PASS<br>Chrome:PASS<br>Firefox:PASS |
| 3 | Click on the survey | Try to click on a survey button at the bottom of the website. It should take you to a survey page | Clicking on the survey button | Taking you to a survey page | Edge:FAIL<br>Chrome:FAIL<br>Firefox:FAIL |

**Test objectives:** To verify application is capable of handling concurrent user interactions without significant degradation in performance.

**HW and SW setup:**

HW setup:
CPU speed: 2.42 HZ
Memory RAM capacity: 16 GB
Hard Drive Storage: 475 GB
The size of the screen: 14 inches

SW setup:
Operating System: Windows 11 Pro
Browser: Chrome, Firefox, and Edge

Page:game
Page: create-game
Page:review

**Feature to be tested:** Performance
**QA test plan:** please see the table below

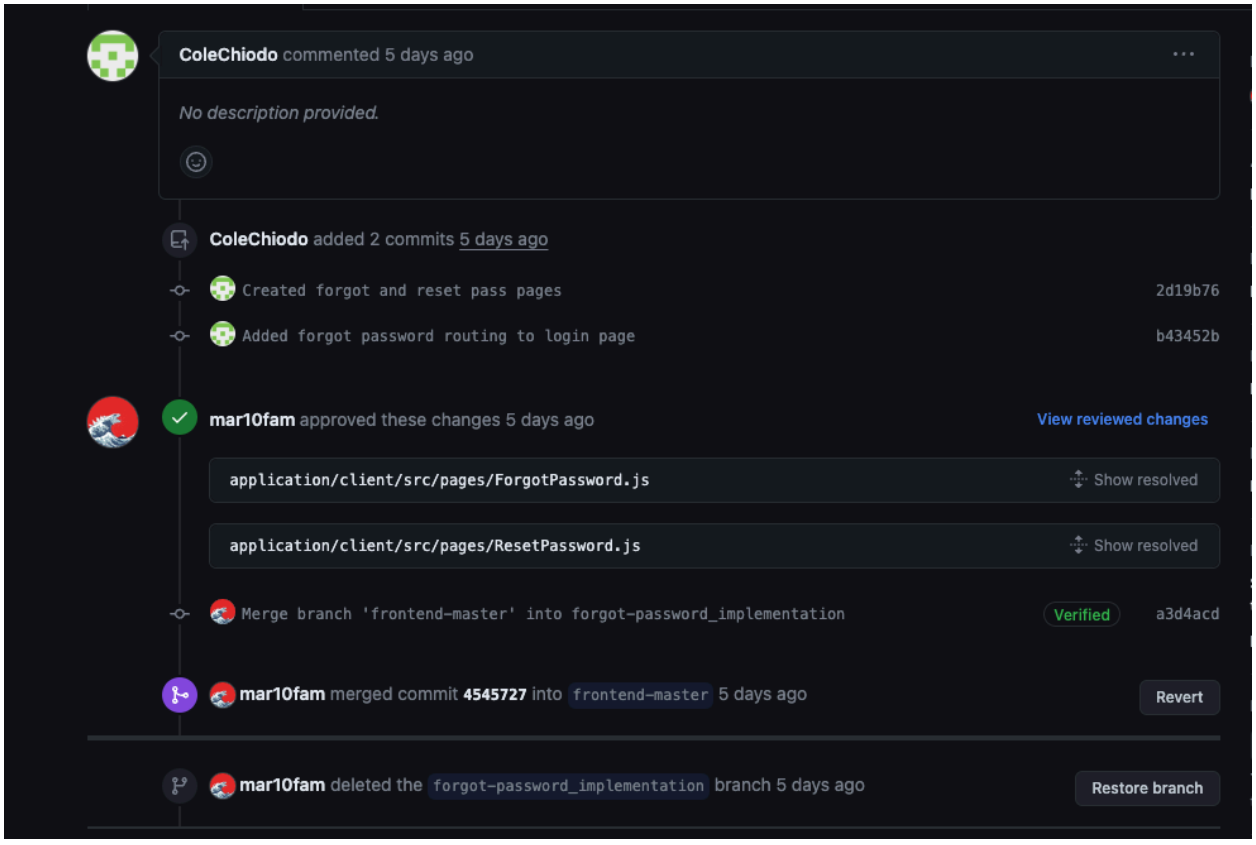
| Test # | Test Title | Test Description | Test Input | Expected Correct Output | Test Results (PASS/FAIL) |
|---|---|---|---|---|---|
| 1 | Joining a team with other users | Click on a game, and you can see two teams. Try to join one of them and do it concurrently with other users. | Try to join a team with other users | They should be able to join a game without any problems | Edge:PASS Chrome:PASS Firefox:PASS |
| 2 | Creating a game with other users | Create a game with other users at the same time. | Try to create a random game with other users. | It should create the games without any problems. | Edge:PASS Chrome:PASS Firefox:PASS |
| 3 | Writing a review with other users | Write a review of other users with other users at the same time. | Try to write a review of a user with other people | They should be able to write reviews without problems | Edge:PASS Chrome:PASS Firefox:PASS |

# 4) Code Review:

## a) Details of team's coding practices:

1. **Use of Components:** Utilize components to modularize and encapsulate functionality. This approach promotes code reusability, readability, and maintainability. By breaking down complex UIs into smaller, manageable components, the code becomes more condensed and easier to understand. Additionally, it encourages a consistent structure across the application.
2. **Headers for Pages and Components:** All pages and components should contain headers that provide essential information such as contributors, a brief description of the purpose and the components used within them. This ensures better understanding of the codebase.
3. **Proper Indentation:** Maintain consistent and proper indentation throughout the codebase for enhanced readability. Clear indentation will help us grasp the code's structure and logic, making it easier to navigate and modify.
4. **Limit CSS Usage, Prefer Tailwind CSS:** Minimize the direct use of CSS and instead leverage Tailwind CSS for styling. Tailwind CSS offers a utility-first approach, allowing us to apply styling directly within HTML markup, resulting in more concise and maintainable code. This approach also reduces the need for writing custom CSS rules and helps achieve a consistent design system.
5. **Line Length Limitation:** Restrict lines of code to a maximum of 150 characters to prevent overly long lines that can negatively impact readability. Breaking lines at appropriate points enhances code clarity and makes it easier to view and understand code in various environments and screen sizes.
6. **Meaningful Function Names:** Choose descriptive and meaningful names for functions that accurately reflect their purpose and behavior. This practice improves code readability and comprehension, making it easier for us to understand the function's role within the application.
7. **Consistent Naming Conventions:** Functions will be written in camelCase (e.g., firstName). This will maintain consistency within the codebase to ensure uniformity and readability.
8. **Exported Pages Naming Convention:** Exported pages should follow PascalCase naming convention (e.g., FirstName) to distinguish them as standalone entities and adhere to standard naming conventions for React components.

# b) Our Code Reviews




# c) External Code Review with Team 06:



**External Code Review - Payment.jsx**

Kirk Ruble <kruble@sfsu.edu>
Sat 2024-05-11 23:05

To:Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>
Cc:Sean Nguyen <snguyen36@sfsu.edu>;Nixxy Dewalt <tdewalt@sfsu.edu>;Vincent Duong <vduong@sfsu.edu>; Gregory Luke Arruiza <garruiza@sfsu.edu>;Prasanna Wagle <pwagle@sfsu.edu>

Greetings Juan Estrada of Team 05,

I believe that you would serve as an effective code reviewer for a recent file we completed named Payment.jsx. Please see the attached zip file below for the code we would like you to review.

Payment.zip

Thank you again for taking the time to review this code we look forward to hearing from you.

Best,
Kirk of Team 06.

## Re: External Code Review - Payment.jsx

Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>
Sun 2024-05-12 14:05
To:Kirk Ruble <kruble@sfsu.edu>

1 attachments (5 KB)
game.controller.ts;

Hi Kirk,

I've reviewed the Payment component you shared. Here are some suggestions:

Functionality and Validation: You've implemented the form inputs and their formatting well. However, ensure all error handling is user-friendly. Specifically, the error alert for the email check could provide guidance on what's wrong instead of just saying the email isn't in the database.

State Management: You're managing state effectively with React hooks. Just a note: consider extracting logic like date calculations and form validations into separate functions or use custom hooks to make the component cleaner and easier to manage.

UX Improvements: When a payment is processed, consider disabling the button or indicating that something is happening to prevent duplicate submissions.

Error Handling: It's good to handle the axios error case, but also consider handling potential errors in your date functions or other logic that might unexpectedly fail.

I'm also attaching a file with some code I need help reviewing. Could you take a look when you have a chance?

Thanks, and looking forward to your thoughts on this!

Best,
Juan Estrada

---

**From:** Kirk Ruble <kruble@sfsu.edu>
**Sent:** Saturday, May 11, 2024 23:05
**To:** Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>
**Cc:** Sean Nguyen <snguyen36@sfsu.edu>; Nixxy Dewalt <tdewalt@sfsu.edu>; Vincent Duong <vduong@sfsu.edu>; Gregory Luke Arruiza <garruiza@sfsu.edu>; Prasanna Wagle <pwagle@sfsu.edu>
**Subject:** External Code Review - Payment.jsx

Greetings Juan Estrada of Team 05,

I believe that you would serve as an effective code reviewer for a recent file we completed named Payment.jsx. Please see the attached zip file below for the code we would like you to review.

Payment.zip

Thank you again for taking the time to review this code we look forward to hearing from you.

## External Code Review - game.controller.ts

Kirk Ruble <kruble@sfsu.edu>
Sun 2024-05-12 15:15
To:Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>

Hello, Juan

I took the time to review the Game Controller file you shared with me. Here are some observations I made:

Code Structure: Your import statements at the top of the file are minimal and neatly organized. Your functions are consistently spaced with no excess code between each function. These practices effectively made the readability of this file easy to view. One improvement I would add to your code structure would be consistent naming in all of your functions. I noticed in your "createGame" and "createGameWithTeams" that you are defining your variables with "snake_case", but throughout the whole file you are using "camelCase".

Code Comments: You have comments in a few of your functions that help with understanding the logic of certain lines of code within them, but there is a lack of consistency throughout the file with providing comments. I would suggest placing comments above each function header in this file to provide insight on how these functions are implemented throughout your project, seeing as I don't know or have access to how these functions interact with your web pages.

Error Handling: It is great that you are handing errors and returning status numbers to your web components. However, make sure that the error messages you are sending back are not ambiguous. In your "joinTeamHandler" function, you are returning "An unknown error occurred" as an error message which could create confusion for a developer or a user if they see this message.

Input Validation: Your import of the joi library for handling input validation is good practice to prevent unexpected behavior and security vulnerabilities. Just make sure that you put it to use as it is declared but never read within this file.

Best Regards,
Kirk of Team 06

# External Code Review with Team 01:



**Code Review**

Lennart Richter <lrichter@sfsu.edu>
Sat 2024-05-11 8:36
To:Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>

1 attachments (2 KB)
likeController.ts;

Hi Juan,

Here is the file for our code review. Please provide any feedback you may have. I look forward to receiving your email.

Thanks,
Lennart (On behalf of Team 1)

## Re: Code Review

Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>
Sun 2024-05-12 13:54
To:Lennart Richter <lrichter@sfsu.edu>

1 attachments (3 KB)
user.controller.ts;

Hi Lenni,

I checked out the LikeController code you sent over. Here are a few pointers:

Constructor: You don't need to assign the services inside the constructor since TypeScript does this automatically when you list them as private in the constructor itself.

Error Handling: The way you check if userId2 is a number is good. For errors, the message you send back could be clearer or more specific based on what went wrong.

Status Codes: Make sure you use CREATED only when something new is actually made. For other errors, 500 Internal Server Error might be more suitable than 422 Unprocessable Entity.

Business Logic: The part where you check if both users liked each other could perhaps go into the LikeService or MatchService. This would keep your controller simpler and just focused on handling the request and response.

Clean Up: If there's any code you're not using anymore, like // const userId1 = 1; it's a good idea to remove it to keep things neat.

I'm also attaching a file with some code I need help reviewing. Could you take a look when you have a chance?

Let me know if you have any questions or need more help with this!

Cheers,

Juan

---

**From:** Lennart Richter <lrichter@sfsu.edu>
**Sent:** Saturday, May 11, 2024 8:36
**To:** Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>
**Subject:** Code Review

Hi Juan,

Here is the file for our code review. Please provide any feedback you may have. I look forward to receiving your email.

Thanks,
Lennart (On behalf of Team 1)

## Code Review

Lennart Richter <lrichter@sfsu.edu>
Tue 2024-05-14 21:30
To:Juan Jose Estrada Zuluaga <jestradazuluaga@sfsu.edu>

Hi Juan,

We've reviewed your code and had a few pieces of feedback.

- there's a typo "const preferencess = await userService.getUserPreferences(req.params.userId);" there's an extra S in "preferences"
- i don't see validation but I'm assuming there is one somewhere else in their code.
- not a big thing but changing the order of parameters in the "try{ const user" to match the order of the "const createUser" or vice versa
- Everything else looks really good

Thanks, Lenni

# 5) Self-check on best practices for security

## Major Assets Being Protected

1. User Data: Personal information such as email, name, date of birth, gender, username, and phone number.
2. Authentication Tokens: Refresh tokens and reset password tokens.
3. User Preferences: User-selected sports and skill levels.

### Password Encryption

We confirm that passwords are encrypted before being stored in the database. The process involves using the bcryptjs library to hash passwords.

**Code Snippet**

```
export const encryptPassword = async (password: string) => {
  const encryptedPassword = await bcrypt.hash(password, 8);
  return encryptedPassword;
};
```

# Input Data Validation

We validate various user inputs to ensure data integrity and security. Below is a list of validated inputs and the corresponding code:

**User Validation**

- Email: Must be a valid email format.
- Password: Custom validation to ensure complexity.
- Name: Required string.
- Role: Must be either USER or ADMIN.
- Date of Birth: Must be a valid date.
- Gender: Required string.
- Username: Required string.
- Phone Number: Required string.

**Code Snippet**

```
const createUser = {
  body: Joi.object().keys({
    email: Joi.string().required().email(),
    password: Joi.string().required().custom(password),
    name: Joi.string().required(),
    role: Joi.string().required().valid(Role.USER, Role.ADMIN),
    // Include new fields in validation schema
    dob: Joi.date().required(),
    gender: Joi.string().required(),
    username: Joi.string().required(),
    phone_number: Joi.string().required(),
  }),
```

**Authentication Validation**

- Email: Must be a valid email format.
- Password: Custom validation to ensure complexity.
- Refresh Token: Required string.
- Reset Password Token: Required string.

**Code Snippet**

```
const register = {
  body: Joi.object().keys({
    email: Joi.string().required().email(),
    password: Joi.string().required().custom(password),
    dob: Joi.date().required(),
    gender: Joi.string().required(),
    username: Joi.string().required(),
    name: Joi.string().required(),
    role:Joi.string().optional(),
    phone_number: Joi.string().required(),

  })
};

const login = {
  body: Joi.object().keys({
    email: Joi.string().required(),
    password: Joi.string().required()
  })
};
```

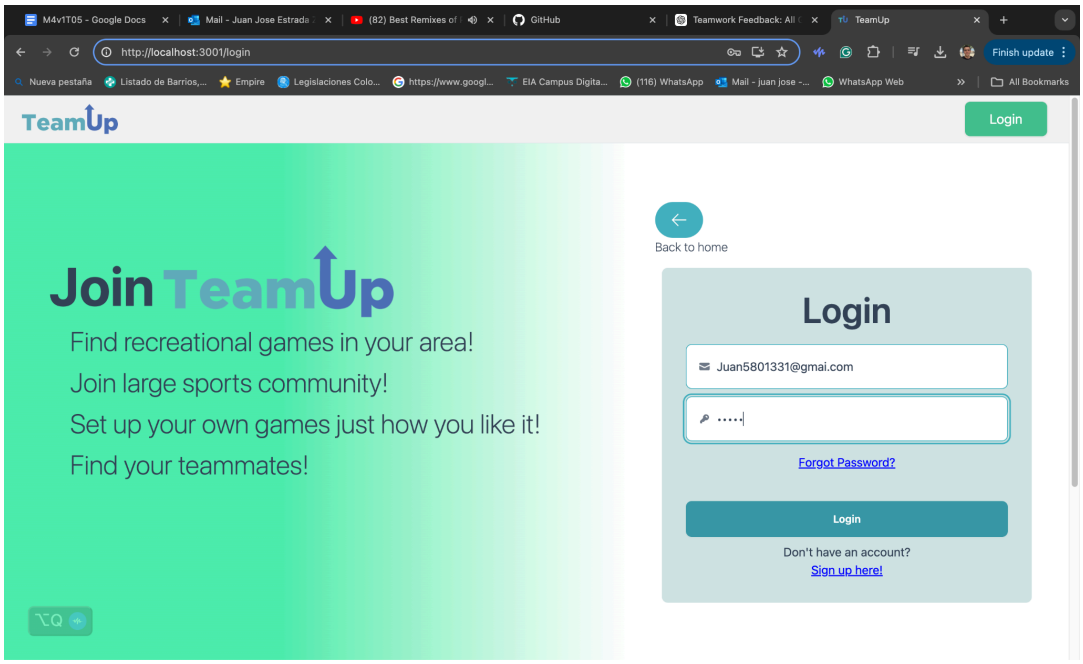
## Search Bar Input Validation

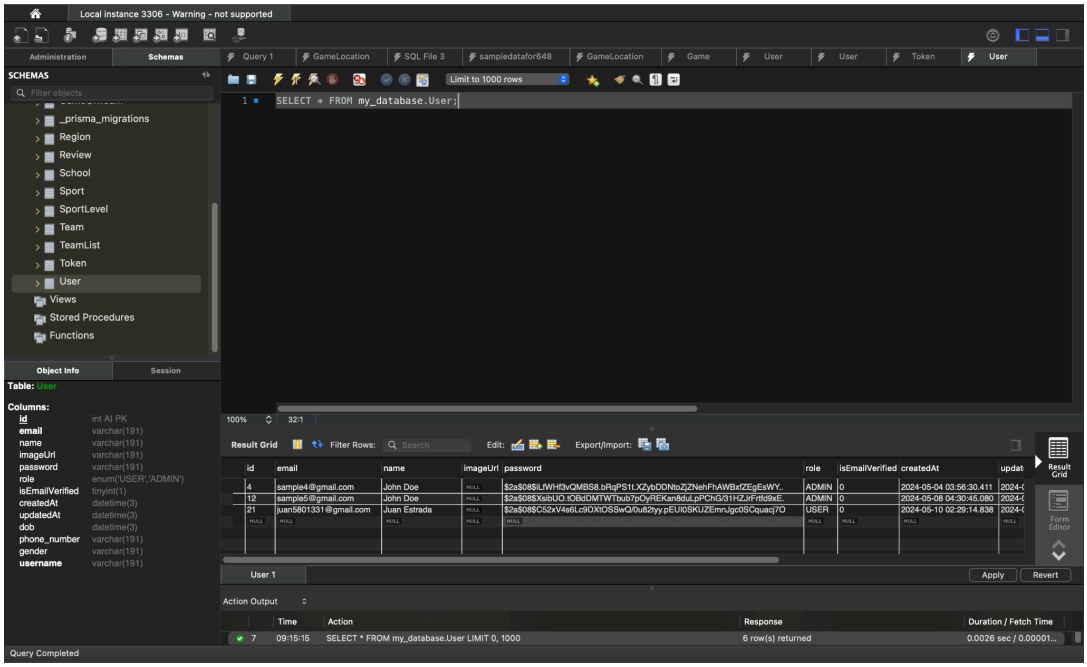
We validate the search bar input to ensure it is a string and prevent potential security issues.

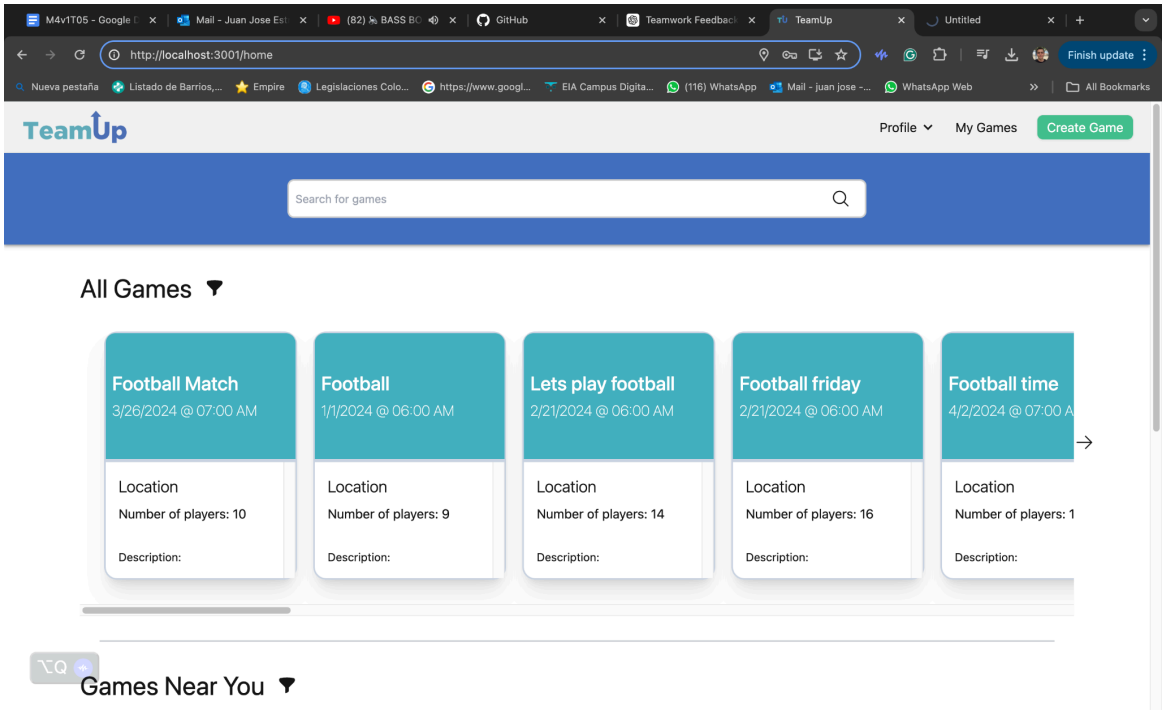
**Example Code**

```
const onSearch = async (searchTerm) => {
    try {
        const searchResults = [];
        for (const sport of selectedSports) {
            const response = await fetch(`${url}/game/search`, {
                method: 'POST',
                body: JSON.stringify({ sport, gameName: searchTerm }),
                headers: {
                    'Content-Type': 'application/json'
                }
            });
            if (!response.ok) {
                throw new Error('Failed to search games');
            }
            const searchData = await response.json();
            searchResults.push(...searchData);
        }
        setGames(searchResults);
    } catch (error) {
        console.error('Error searching games: ', error);
    }
};
```

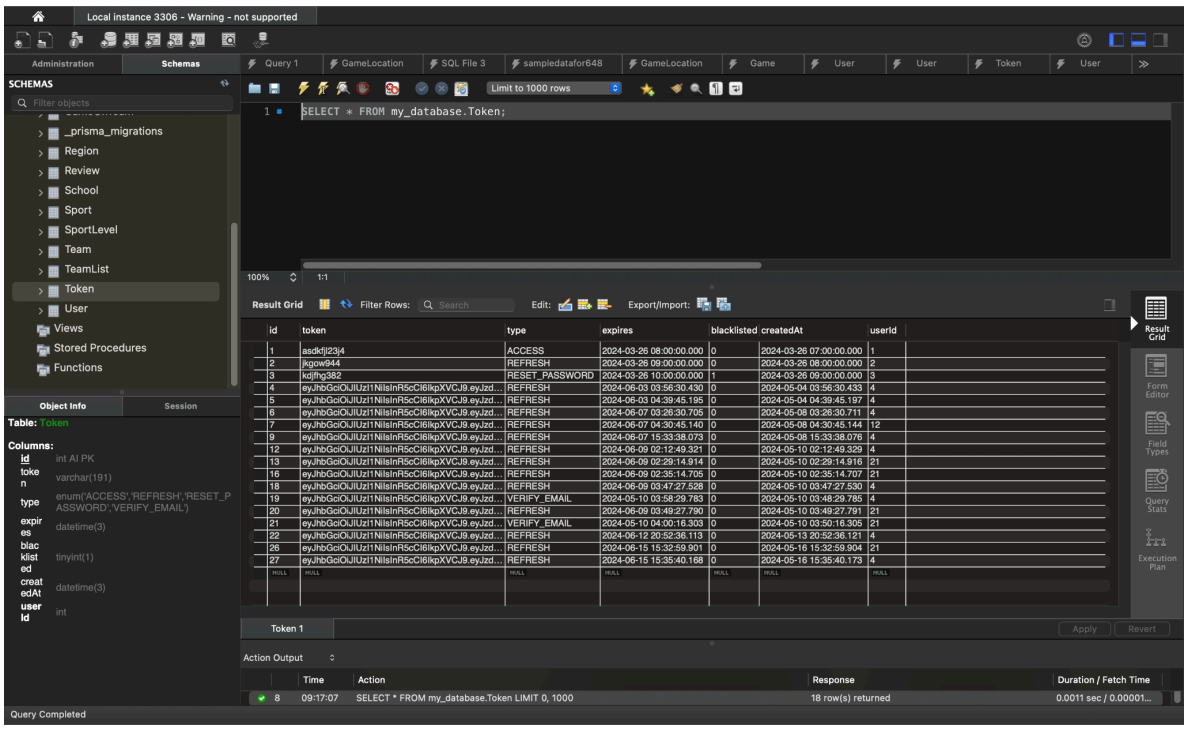
## User logins:


TeamUp
Login
Join TeamUp
Find recreational games in your area!
Join large sports community!
Set up your own games just how you like it!
Find your teammates!
Back to home
Login
Juan5801331@gmai.com
Forgot Password?
Login
Don't have an account?
Sign up here!


Local instance 3306 - Warning - not supported
SCHEMAS
_prisma_migrations
Region
Review
School
Sport
SportLevel
Team
TeamList
Token
User
Views
Stored Procedures
Functions
SELECT * FROM my_database.User;
Result Grid
Action Output
Query Completed




## User is then given token for Session:



# 6) Self-check: Adherence to original Non-functional specs

## Reliability:

- The system shall be available 99.9% of the time during peak usage hours. DONE
- The application shall be resilient to server failures and able to recover gracefully without data loss. DONE

## Response Time:

- The system shall respond to user interactions within 2 seconds for browsing games and tournaments. DONE
- User actions such as joining a game or registering for a tournament shall complete within 5 seconds. ON TRACK

## Hardware and Networking Requirements:

- The application shall be hosted on servers with sufficient processing power and memory to handle concurrent user requests. DONE
- Network bandwidth shall be adequate to support simultaneous interactions from multiple users. DONE

## Usability Requirements:

- The user interface shall be intuitive and easy to navigate for users with basic computer literacy. DONE
- User interactions and workflows shall be consistent across different sections of the application. DONE

## Privacy:

- User data collected shall include name, email address, and sports preferences. DONE
- Data collected shall only be used for facilitating game and tournament management and shall not be shared with third parties without user consent. DONE

## Compatibility:

- The application shall be compatible with major web browsers such as Chrome, Firefox, Safari, and Edge. DONE
- Responsive design principles shall be implemented to ensure optimal viewing and usability across desktop and mobile devices.DONE

## Media Content:

- Images and media files uploaded by users shall adhere to specified formats and size limits to ensure efficient storage and retrieval. DONE
- Accepted file formats and size limits shall be communicated to users during the upload process. ON TRACK

## Performance:

- The application shall be capable of handling concurrent user interactions without significant degradation in performance. DONE
- Database queries and server-side operations shall be optimized to minimize latency and ensure smooth user experience.DONE

## Licensing and Legal:

- The application shall comply with relevant laws and regulations governing data privacy, user consent, and intellectual property rights. DONE
- Proper licensing for third-party libraries and software components used in the development shall be obtained and documented. ON TRACK

# Training and Support:

- Users with a high-school diploma, after 1 hour of training, shall be able to navigate and use core features of the application. DONE
- Support documentation and resources shall be provided to assist users in troubleshooting common issues and accessing help when needed.ON TRACK

# 7. Detailed list of contributions

| | | |
|---|---|---|
| Juan Estrada | 9 | • Self-check: Adherence to original Non-functional specs<br>• API to get user bio, profile image, name, username, reviews and achievements<br>• POST API detached from a team<br>• POST API GameLocation Review<br>• GET review API by username<br>• POST API review user by username<br>• Add bio attribute for user entity<br>• API to post a bio of the user |
| Kotaro Iwanaga | 9.5 | • QA test plan<br>• API to post bio is implemented.<br>• GET API GameLocation list<br>• GET API team1 & team2 by game ID<br>• Create dummy data 30 locations |
| Cole Chiodo | 9 | • Frontend API implementation (Get user by ID profile page)<br>• Product Summary<br>• Frontend API account email verification.<br>• Frontend API implementation (Forgot password & Reset password) |

| | | |
|---|---|---|
| Martin Pham | 9 | • Implement search filter by day and time<br>• Code Review<br>• Implement Search Nearby games<br>• Implement adding, updating, removing, retrieving preferences.<br>• Implement geolocation to track user latitude and longitude |
| Jaycee Lorenzo | 9 | • Usability test plan<br>• Frontend API implementation Logout<br>• Detach API frontend implementation<br>• Implement UX/UI corrections from the last milestone.<br>• Create a Join Game component and implement it with its API<br>• Create User profile page<br>• Implement POST and GET API user review |
| Areeb abbasi | 9 | • Implement s3 bucket to store user profile & game location images<br>• Function to send email notification<br>• Document deployment and deploy app in one EC2 instace<br><br>• Self-check on best practices for security |

# Team Member Contributions

| Name | Grade | Signature | Comment |
|---|---|---|---|
| Juan Estrada | 9 | Juan Estrada | Juan Estrada served as the team lead for our group, demonstrating leadership and organizational skills. He was responsible for creating the project workflow, ensuring everyone was aware of their tasks, and leading our meetings. Juan's experience in the industry and participation in hackathons provided him with a broad understanding of software product development, which was very helpful for this class project. His ability to guide the team, keep us on track, and offer insights from his experiences greatly contributed to our overall success. |
| Kotaro Iwanaga | 9 |  | Kotaro was incredibly helpful and reliable throughout the project. Despite starting with no experience in backend development, he quickly learned and made substantial contributions. He was always punctual with his tasks and responsible in attending meetings. His work on the database architecture, EER, ERD, and various APIs, including those for game listing and user preferences, was excellent. Kotaro's dedication and ability to adapt were crucial to our project's success. |
| Cole Chiodo | 9 | Cole Chiodo | Cole was very responsible and consistent with his tasks and meeting attendance. His attention to detail in developing pages and formatting milestone documents ensured high-quality outputs. Although he initially lacked experience with React, he diligently completed his tasks, including front-end API implementations for user profiles, email verification, and password management. Cole's reliability and commitment were greatly appreciated. |
| Martin Pham | 9 |  | Martin was meticulous in his work and helped identify and address requirements for our milestones that we might have |

| | | | |
|---|---|---|---|
| | | | overlooked. He was responsible with his tasks and meetings, effectively managing use cases, wireframes, and the implementation of search filters and geolocation. His overall contributions were significant. Martin's attention to detail and commitment were vital to our project's progress. |
| Jaycee Lorenzo | 9 | Jaycee Lorenzo | Jaycee was crucial for the development of our app, especially on the frontend. He led the frontend tasks, creating wireframes, the game page, and various components. His enthusiasm for our project was clear, treating it as a real product for users rather than just a school project. Jaycee's work on implementing UX/UI corrections, creating the Join Game component, and developing the user profile page was great. His leadership and positive attitude significantly motivated our team. |
| Areeb abbasi | 8 | Areeb Abbasi | Areeb brought valuable technical experience to our project, especially in deployment and backend development. Despite occasional challenges in attending meetings and meeting deadlines, his contributions were essential. He played a key role in setting up the backend and resolving deployment issues and backend bugs. Areeb's expertise with the S3 bucket for storing user profile and game location images, and his function to send email notifications, were important assets to our team. |

# Post analysis

Main Challenges:

- Time Management:
  We often missed our deadlines and finished tasks just before they were due. This happened because we didn’t manage our time well and weren't as productive as we could have been.

- Project Complexity:
  Our website ended up being more complicated than planned. We decided to cover many sports, which made the project harder to manage compared to if we had focused on just one sport.

- Story: When we were submitting Milestone 3, the auto-deployment of the app caused us a lot of trouble. We had difficulty finding the mistake. To solve this issue, we held a meeting to discuss recent changes to the code that might have caused the problem. We identified that the issue was related to Next.js but still couldn’t pinpoint the exact problem. As we were falling behind schedule, we reorganized tasks so that Areeb, the person in charge of solving this issue, could focus solely on it. We initially thought the problem was due to compatibility with AWS and spent $20 deploying on Vercel. However, after deploying on this new server provider, we encountered the same error. After carefully reviewing each file of our current code, we discovered that macOS doesn't consider capitalization in file imports, while the Ubuntu server does. This mismatch caused the code to break. We learned how to target bugs effectively and manage our time to solve them.

What We Would Do Differently:

- Shorter Meetings:
  At the start, our meetings were an hour long. Shorter meetings (less than 30 minutes) would have given us more time to work on the project itself.

- Better Communication:
  We only discussed what we were working on during meetings. If we had shared daily updates on Discord, we could have kept better track of each other's progress and managed our tasks better.

- Focus on One Sport:

Instead of creating a website for all sports, we should have focused only on soccer. This would have made the project simpler and allowed us to spend more time on special features for the website.

- Choosing the Right Framework:
  We spent too much time switching between Next.js and React. If we had decided on one framework from the start, it would have saved us time and made our planning more effective.

- More Time for Deployment:
  We wish we had spent more time or started earlier on deploying our site. This would have made the process smoother and less rushed.

- Push for More Requirements:
  We wish we had pushed harder to meet more of our P2 requirements. Achieving these goals would have made our project more complete and impressive.

Personal Experience:

During the project, we learned how to work well with people who have different skills and ideas. This helped us find new ways to solve problems. Some team members also acted as mentors, which was beneficial for both the mentors and the ones they were helping.

We learned how to depend on and trust others. There were times when we had to trust that others would complete their tasks so we could finish ours. Trust is important because it not only keeps everyone accountable but also allows us to focus on our own tasks instead of worrying about someone else's.