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:

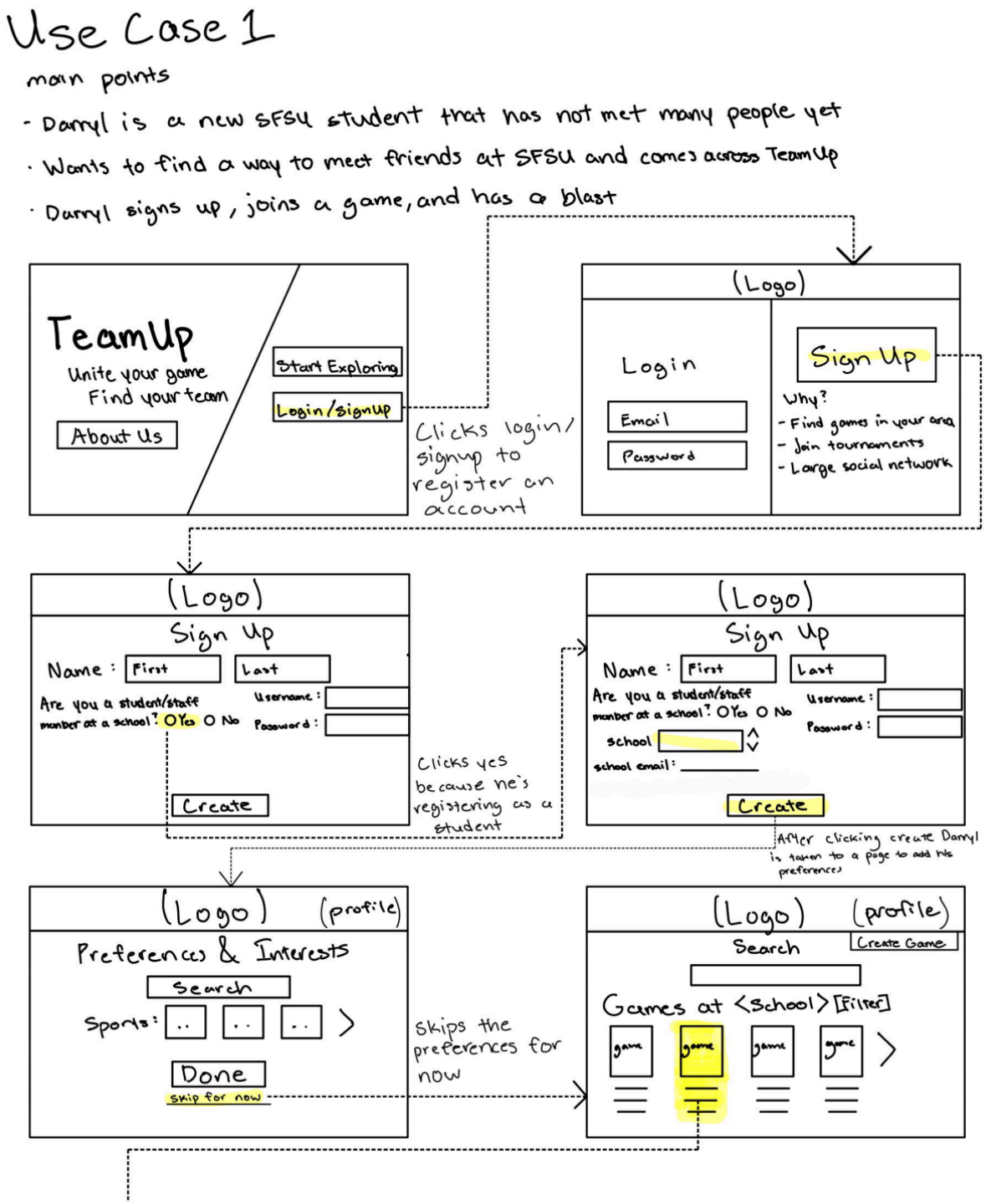
38. Users shall have privacy settings to control who can see their profile and game activity.
39. Users shall receive real-time updates about game changes (like location changes, time adjustments).
40. Users shall see leaderboards for various metrics like most games played,
41. Games shall be able to 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.

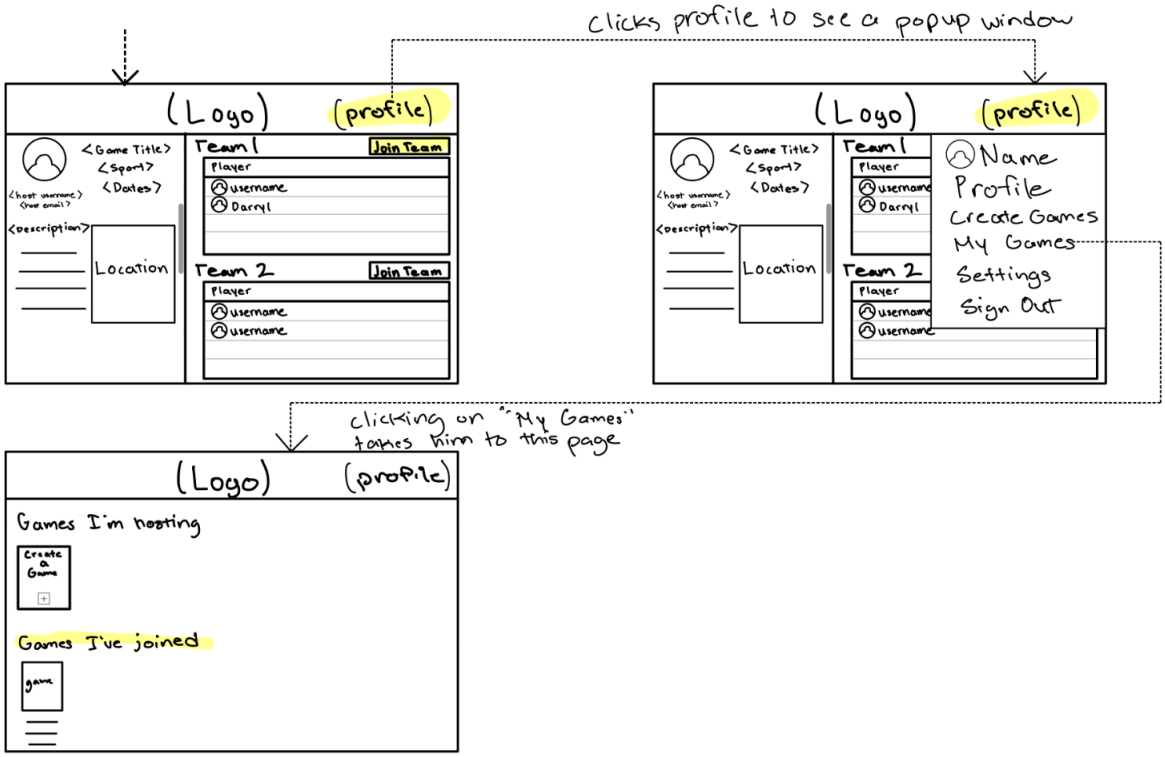
44. Game listings shall include weather forecasts for the scheduled time and location.

Tournament:

45. Tournaments shall offer options to livestream games.

3. UI Mockups and Storyboards

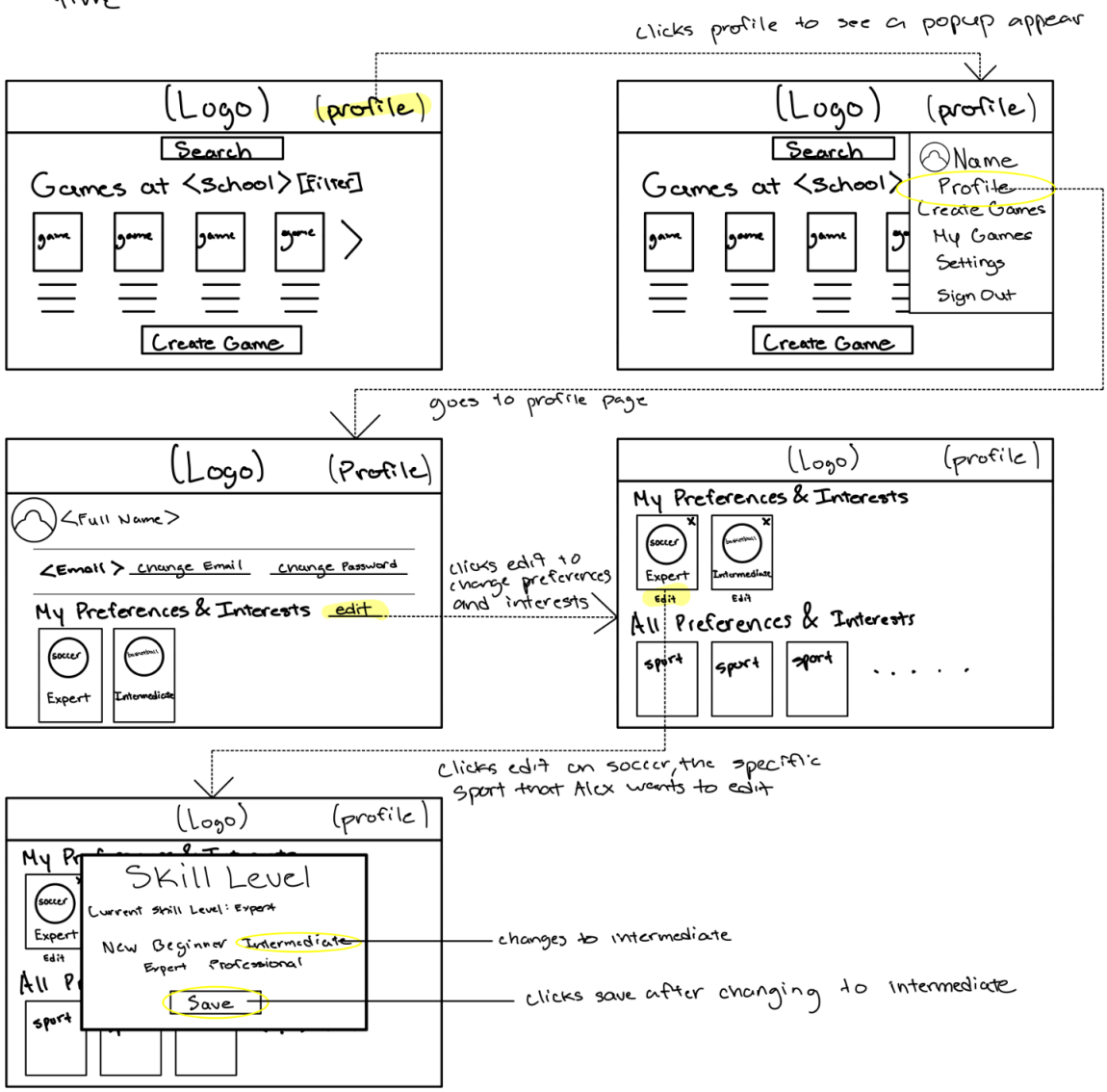



clicks profile to see a popup window
(Logo)
(profile)
<Game Title>
<Sport>
<Dates>
<host username>
<host email>
<description>
Location
Team 1
Join Team
Player
username
Darryl
Team 2
Join Team
Player
username
username
Name
Profile
Create Games
My Games
Settings
Sign Out
clicking on "My Games" takes him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

Use Case 2

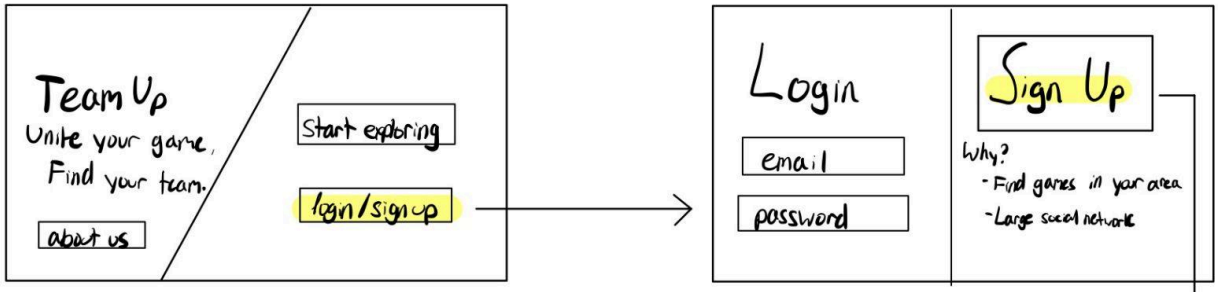
main points:

- 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

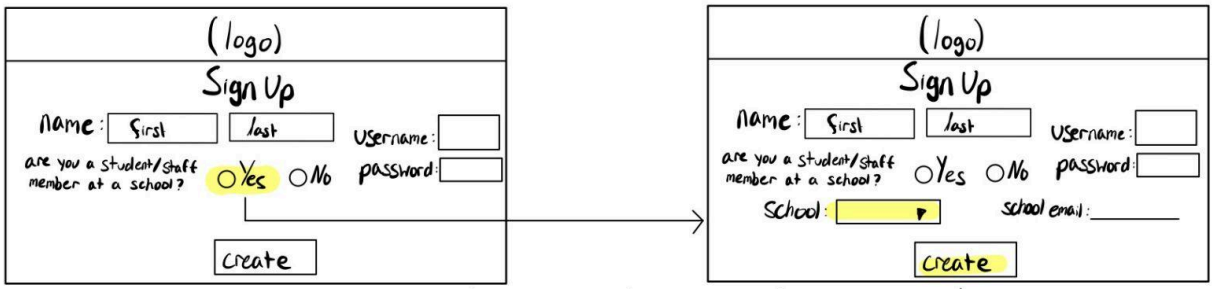


Use Case 3

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.

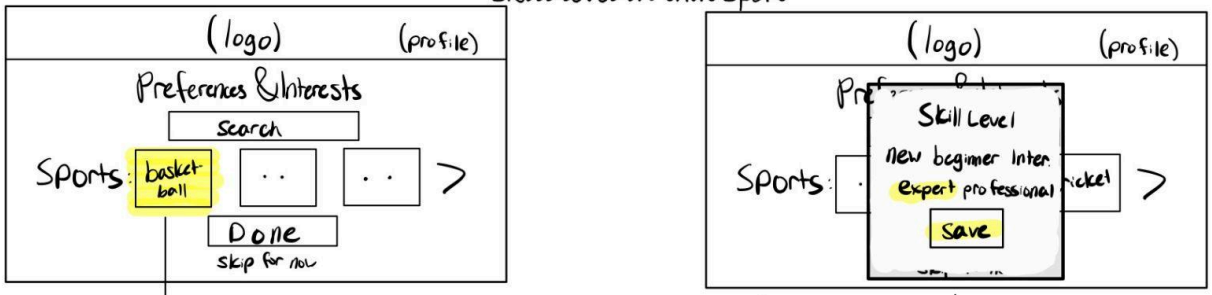


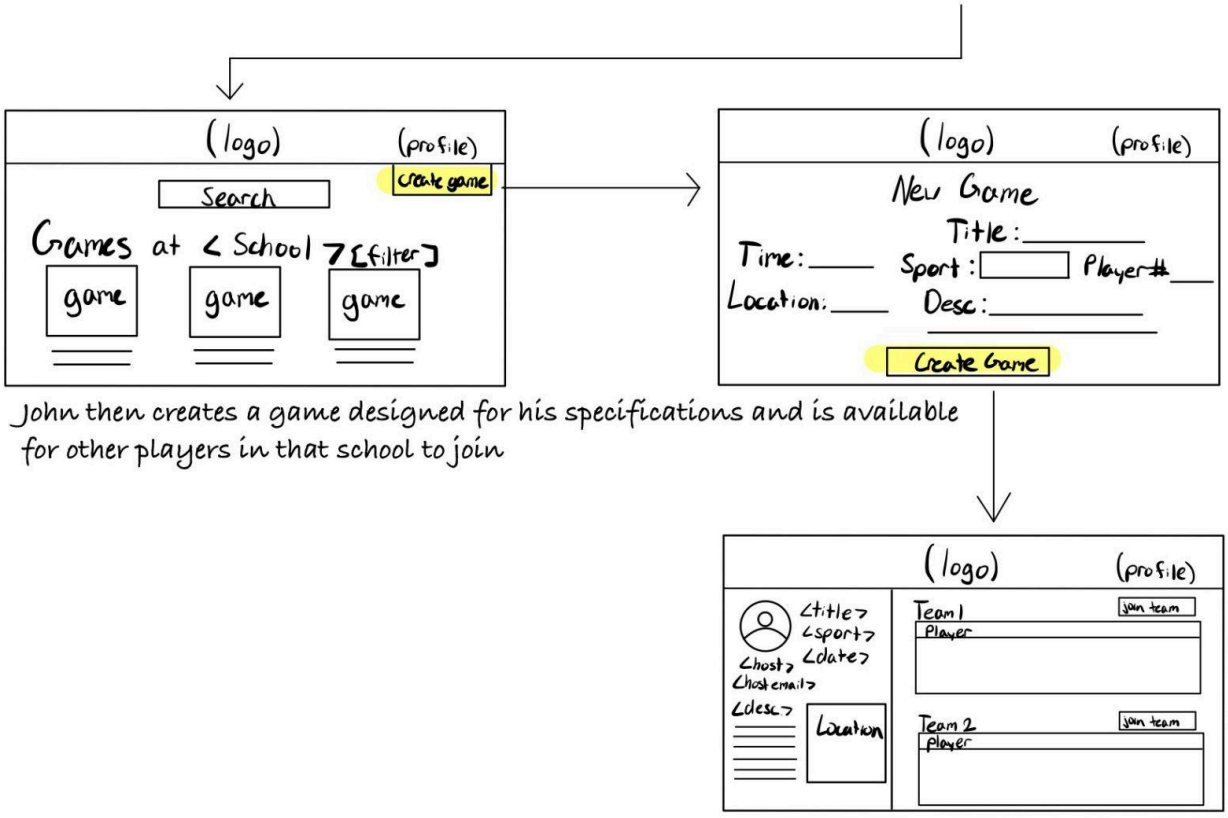
John is greeted by the title screen and clicks login/sign up to create an account



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

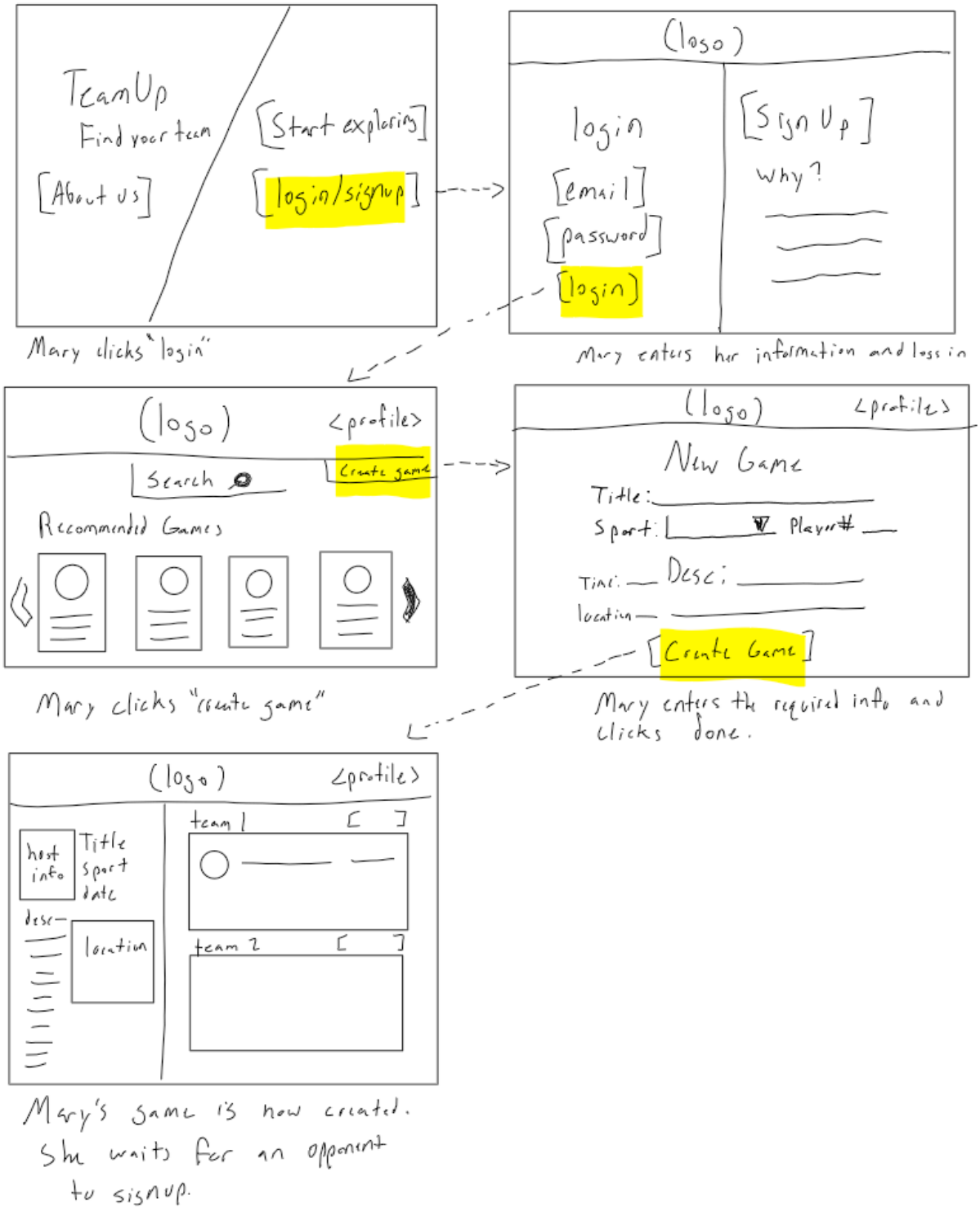
John finds his preferred sport and states his skill level in that sport




(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

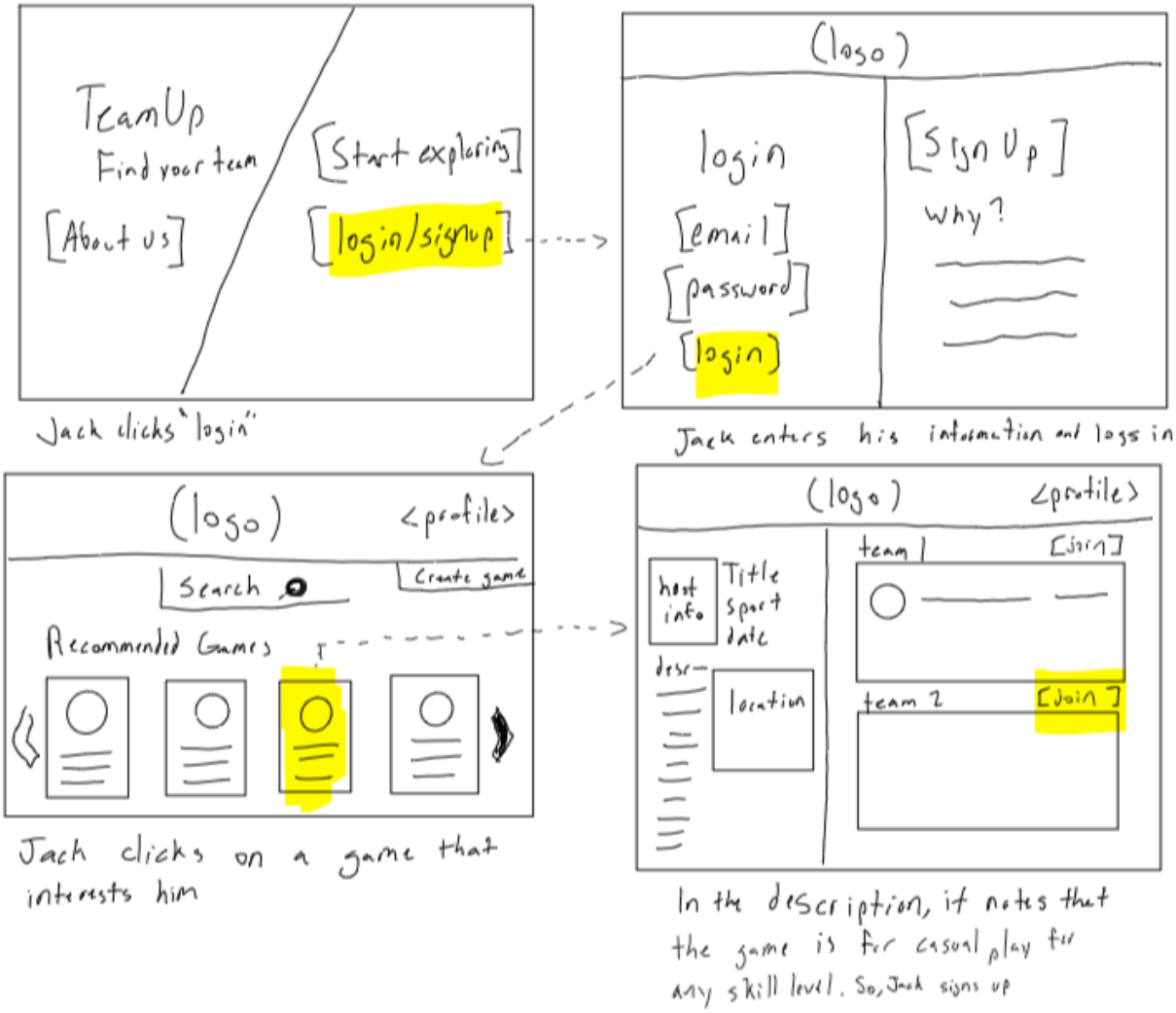
Use Case 4

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



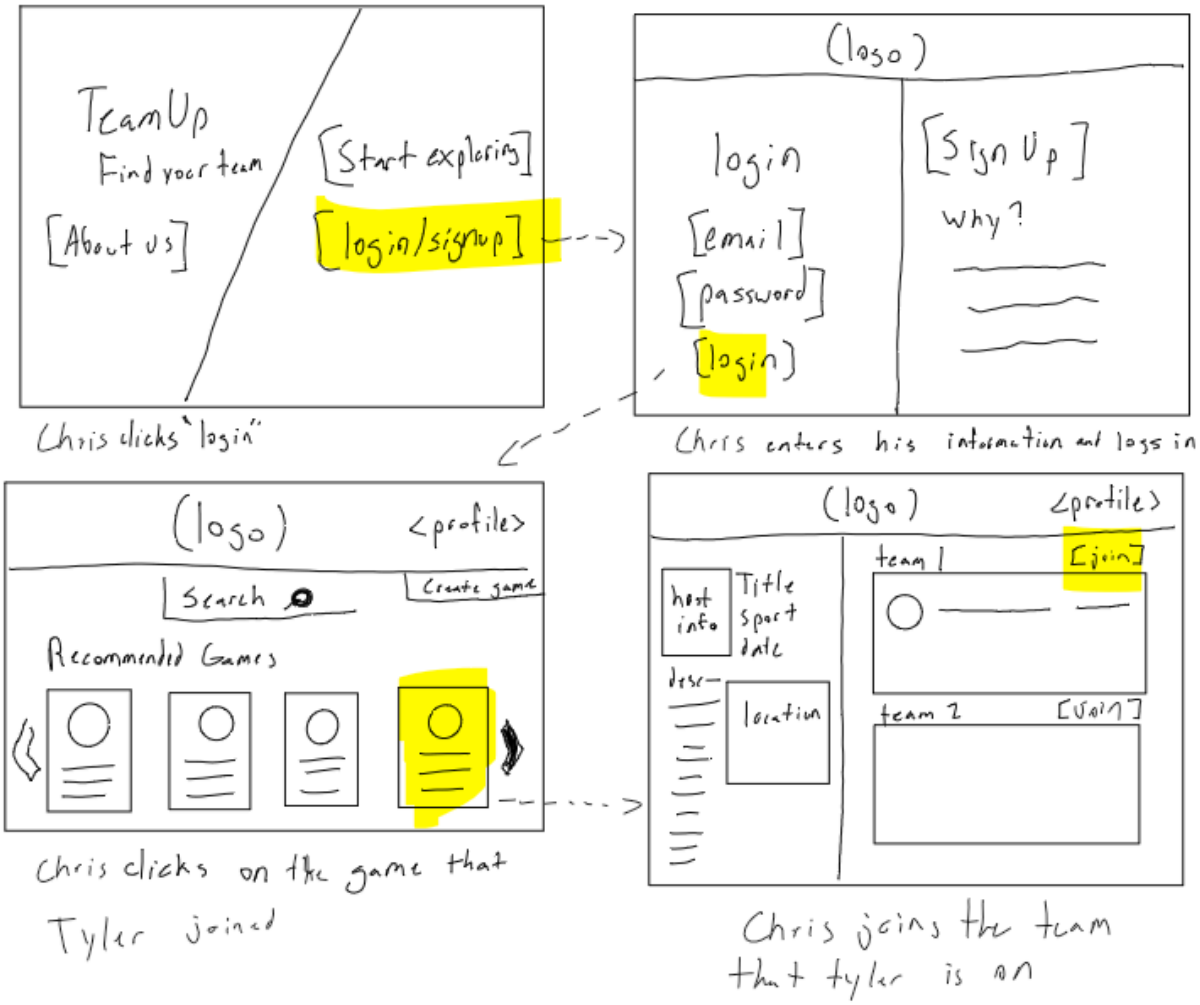
Use Case 5

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



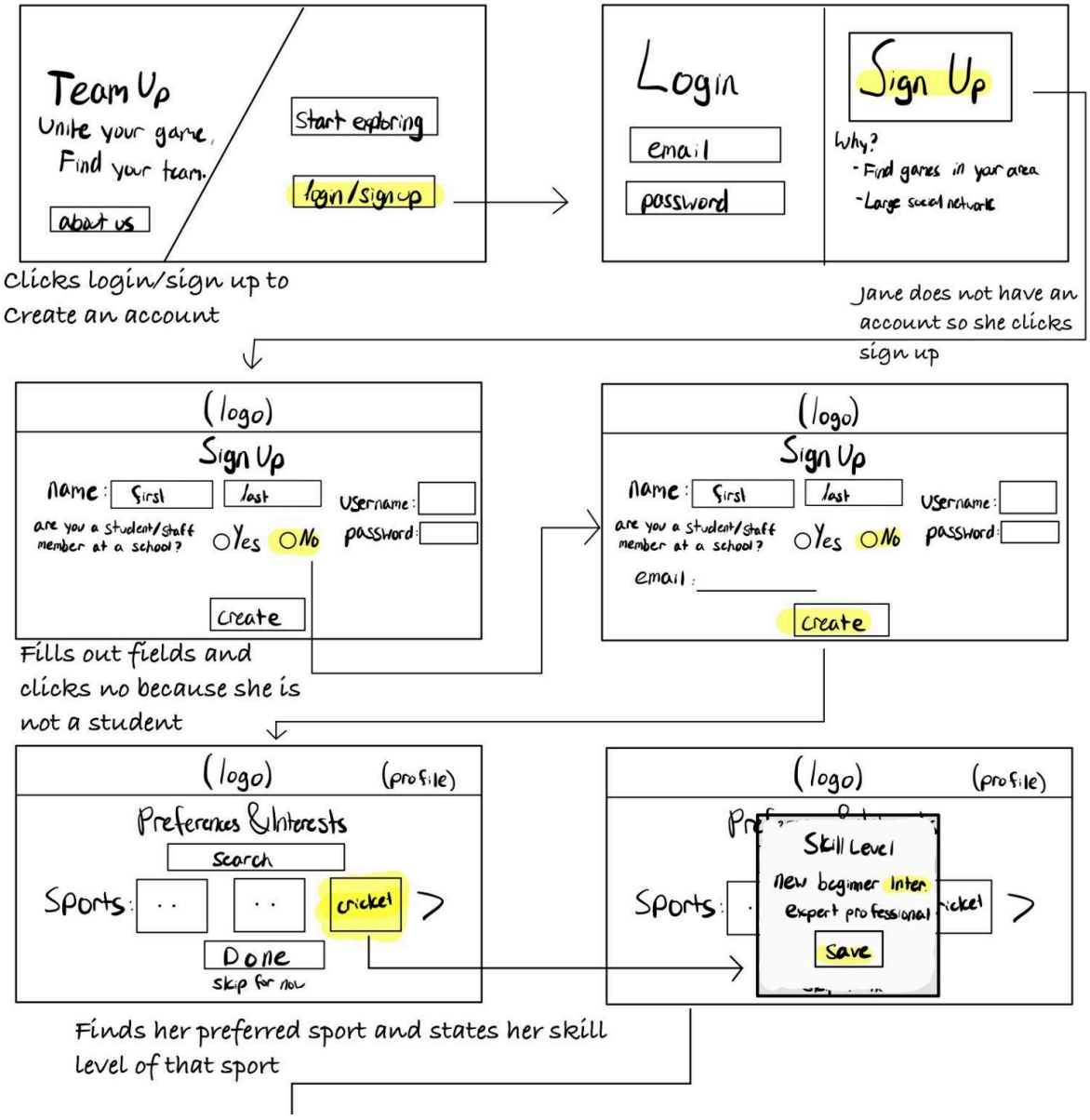
Use case 6

- 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

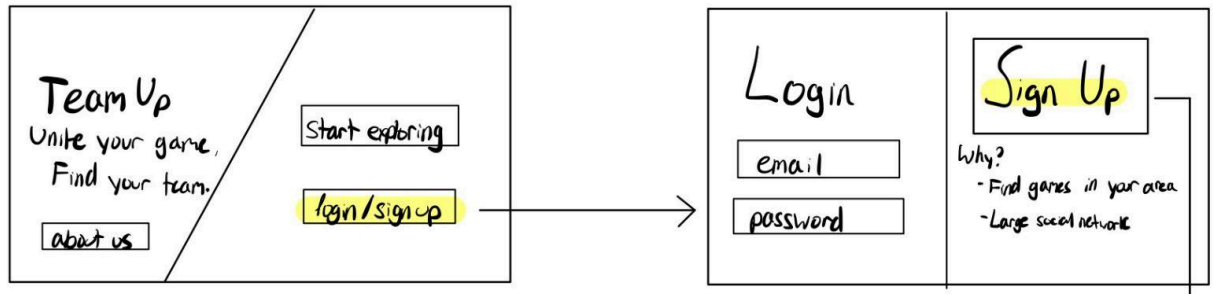
main points:
- Jane moves to San Francisco.
- Jane plays cricket, but cannot find any games nearby.
- Jane comes across TeamUp online and creates an account.
- Jane is now able to find cricket matches in the city.



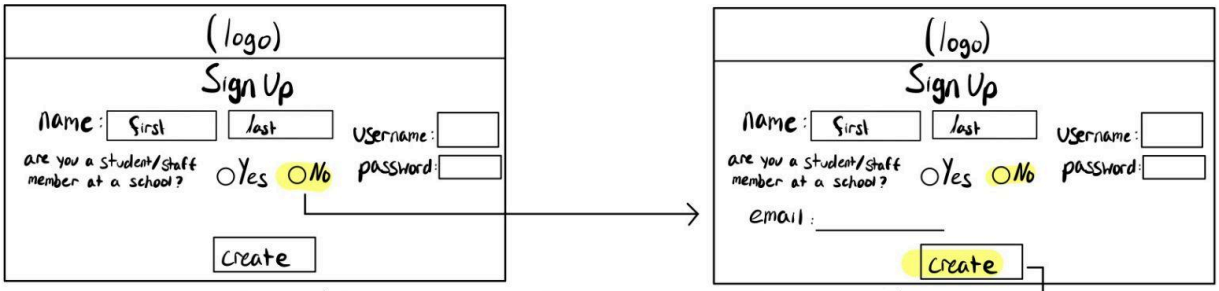
Use Case 8

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

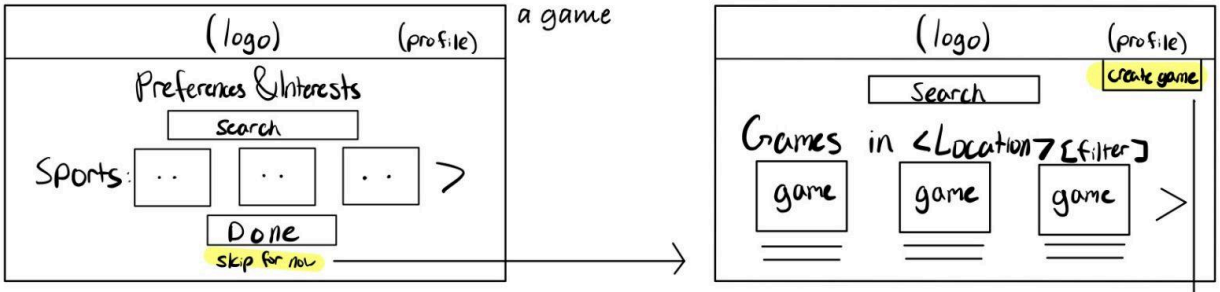


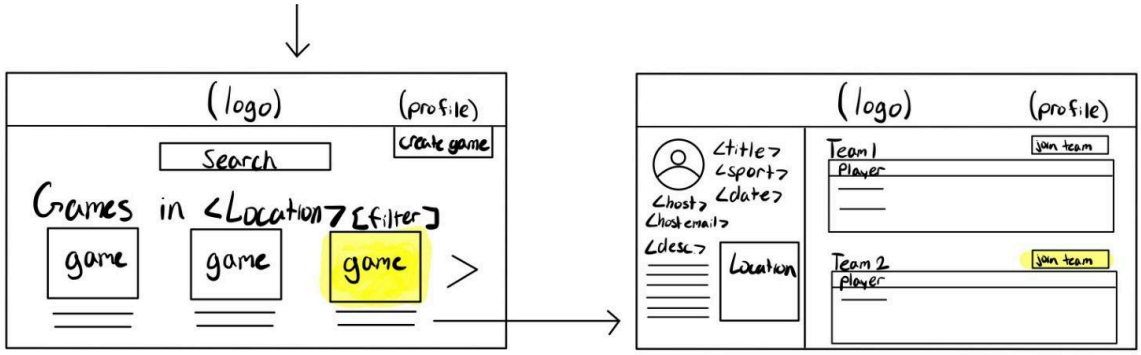
Trevor and Noah click login/sign up to create their accounts



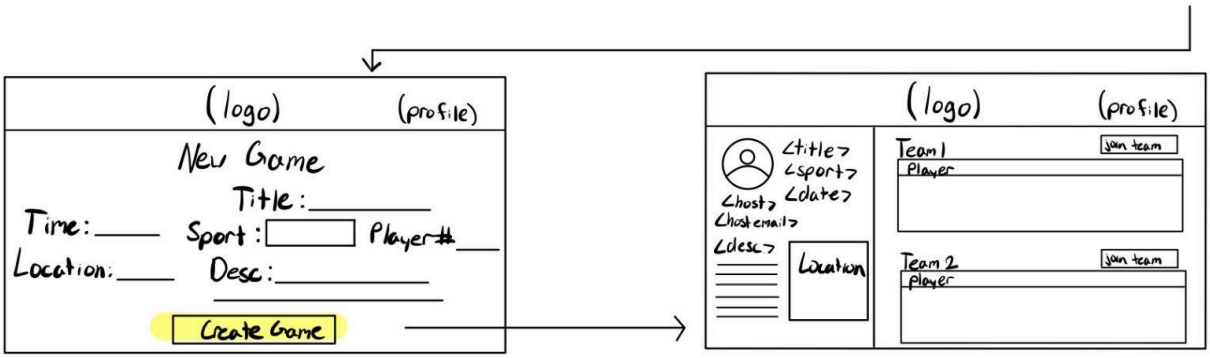
They click "no" because they are not students and fill out the necessary information

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.

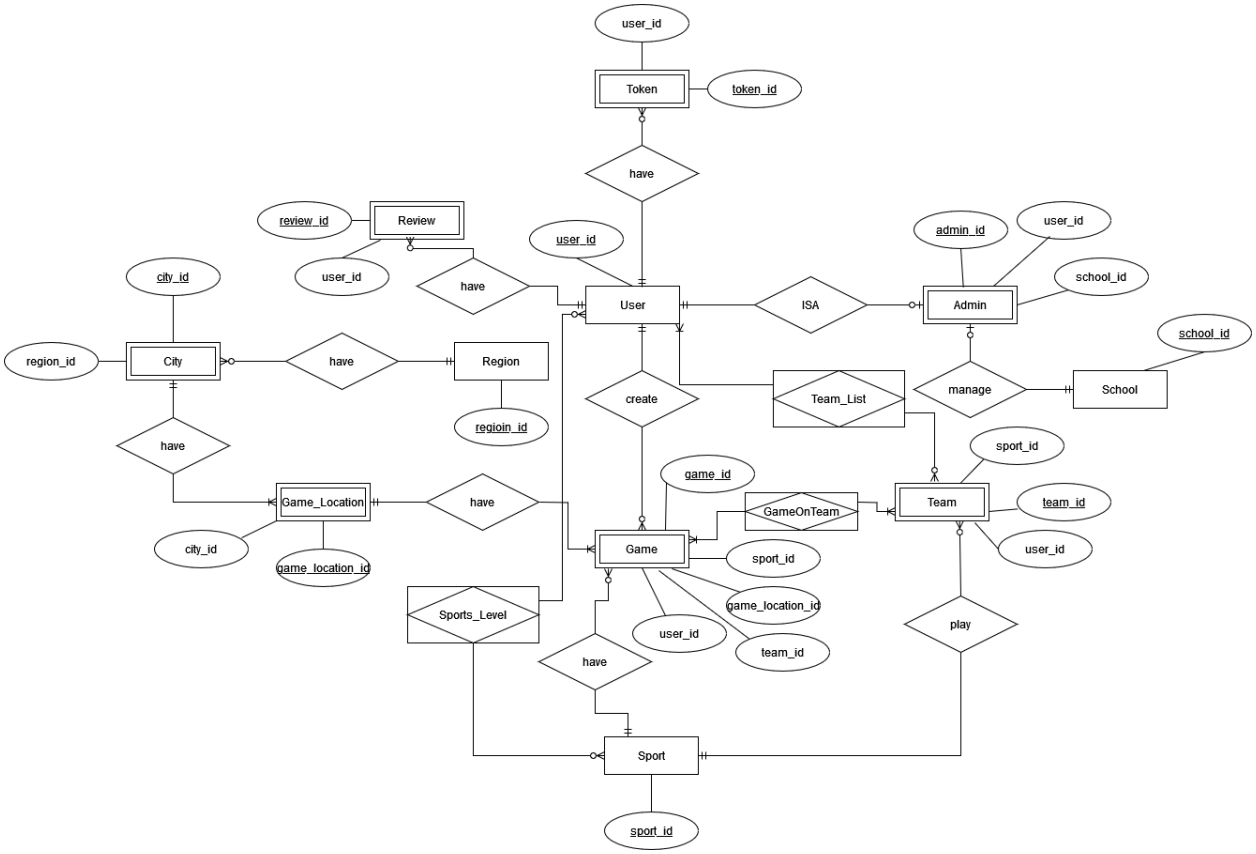
10. **Review (Weak)**
    - review_id: PK, numeric
    - user_id: FK, numeric
    - rating: alphanumeric
    - description: alphanumeric

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

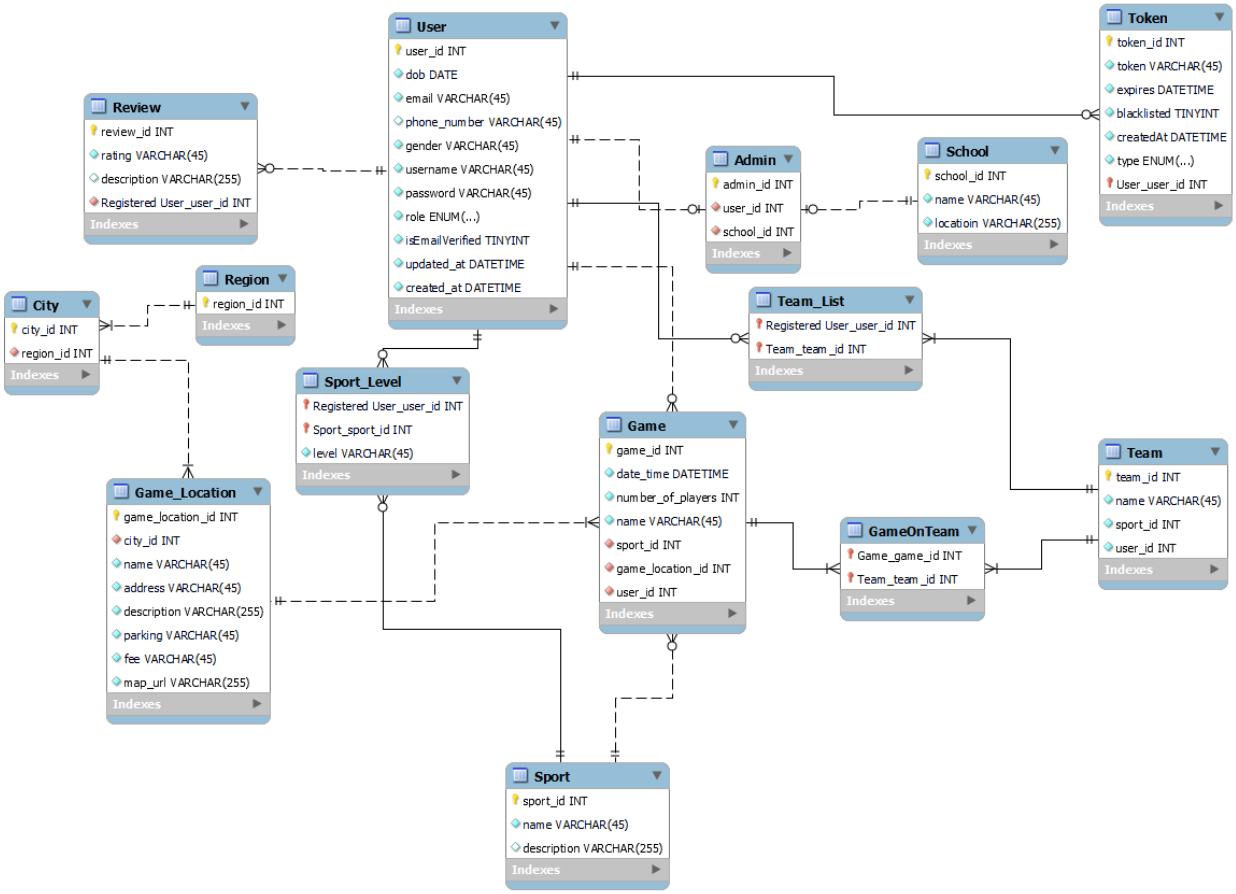
11. **Token (Weak)**
    - token_id: PK
    - user_id: FK
    - token: alphanumeric
    - type: enum
    - expires: date
    - blacklisted: boolean
    - createdAt: date

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
create
Team_List
manage
School
school_id
have
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
play
user_id
team_id
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.

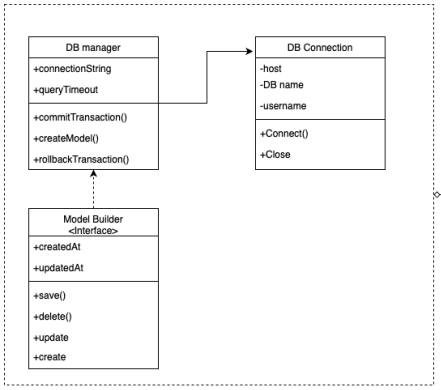
New Framework:

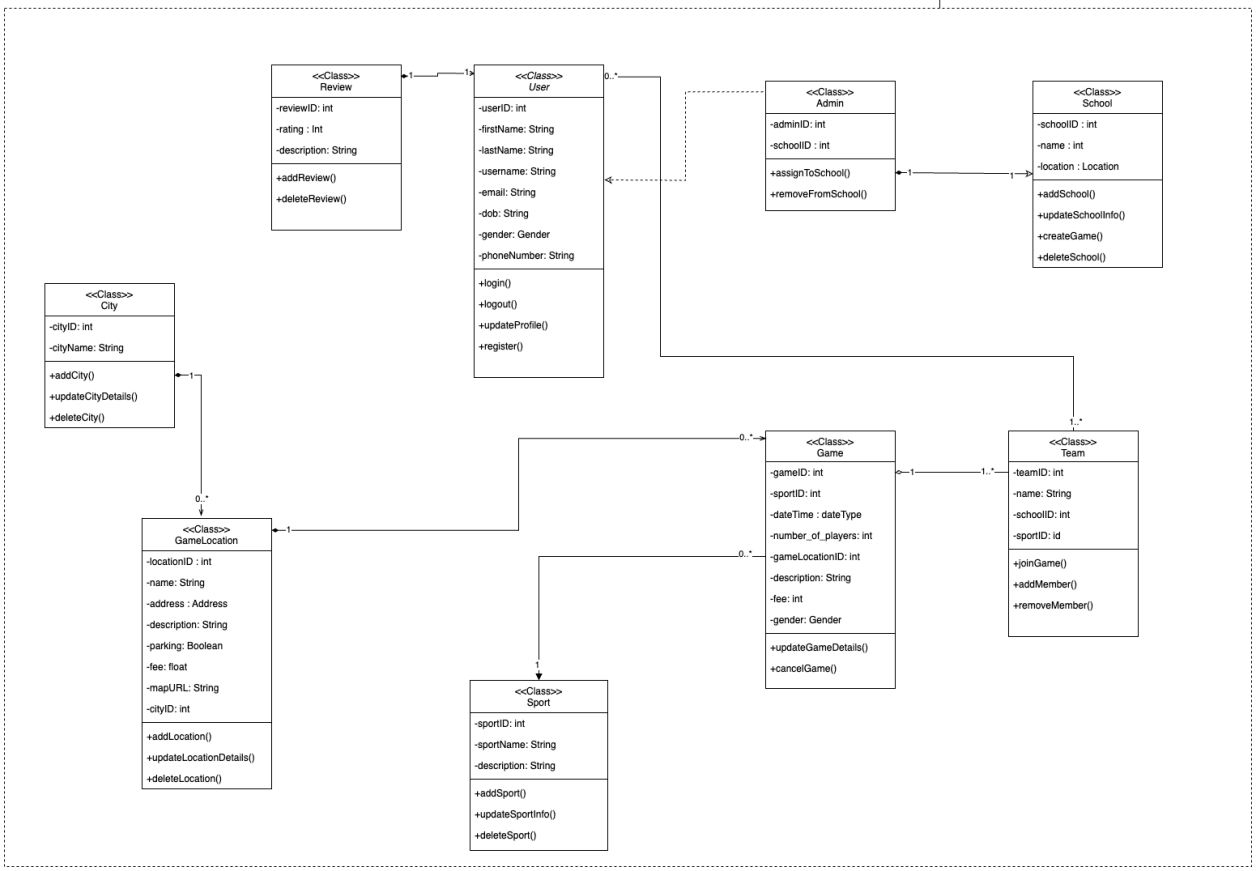
One new tool that we have decided to use is Prisma, because it makes working with databases much easier and more efficient. Prisma is great because it lets us handle databases in a simpler, safer, and more developer-friendly way. We chose it because we want our development process to be strong, quick,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

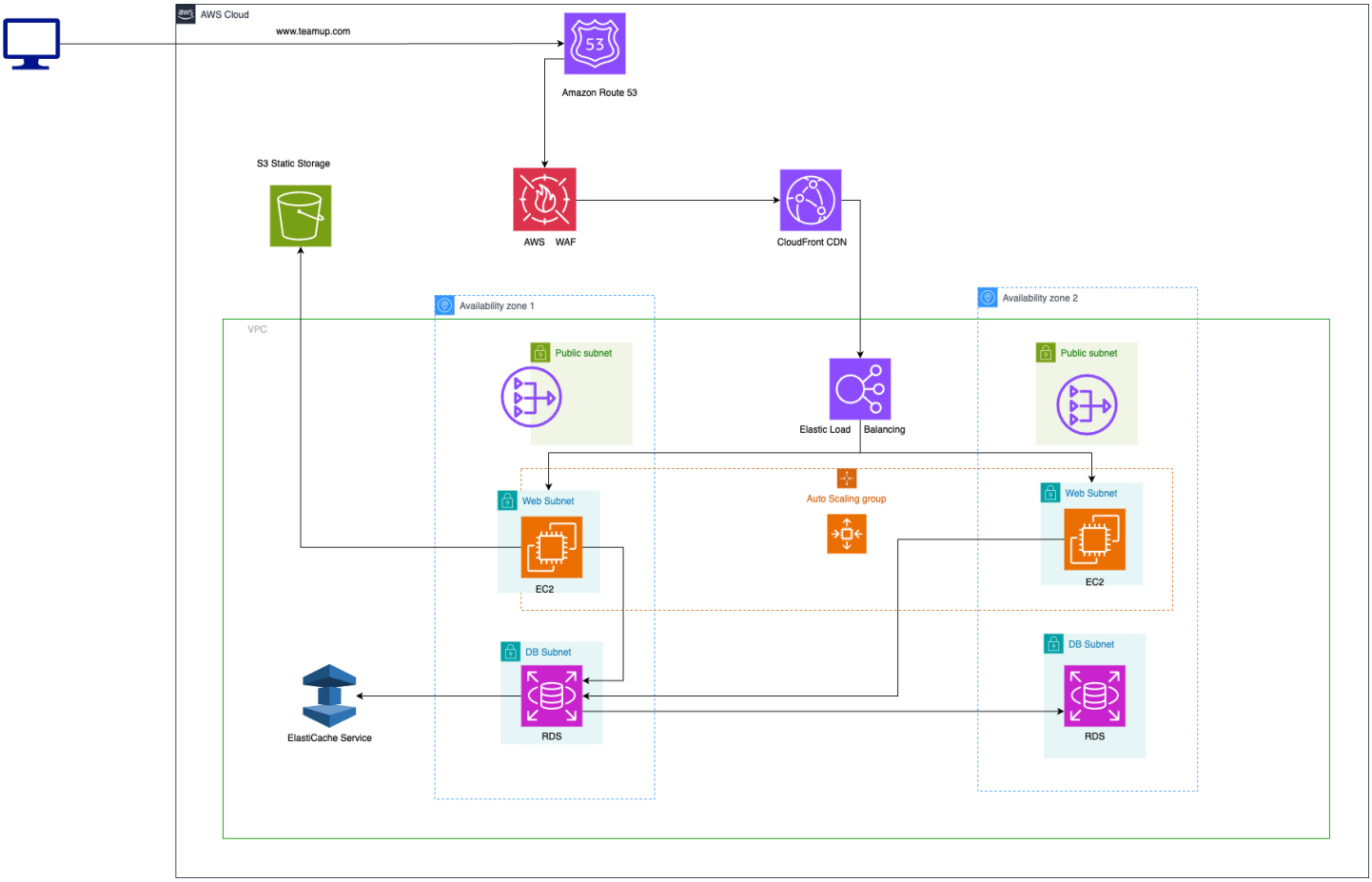
We've adopted design patterns that are conducive to modularity. For instance, the 'DB manager' class centralizes data management functions, offering a single point of interaction with the database through a well-defined interface. This encapsulation simplifies maintenance and future upgrades to data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

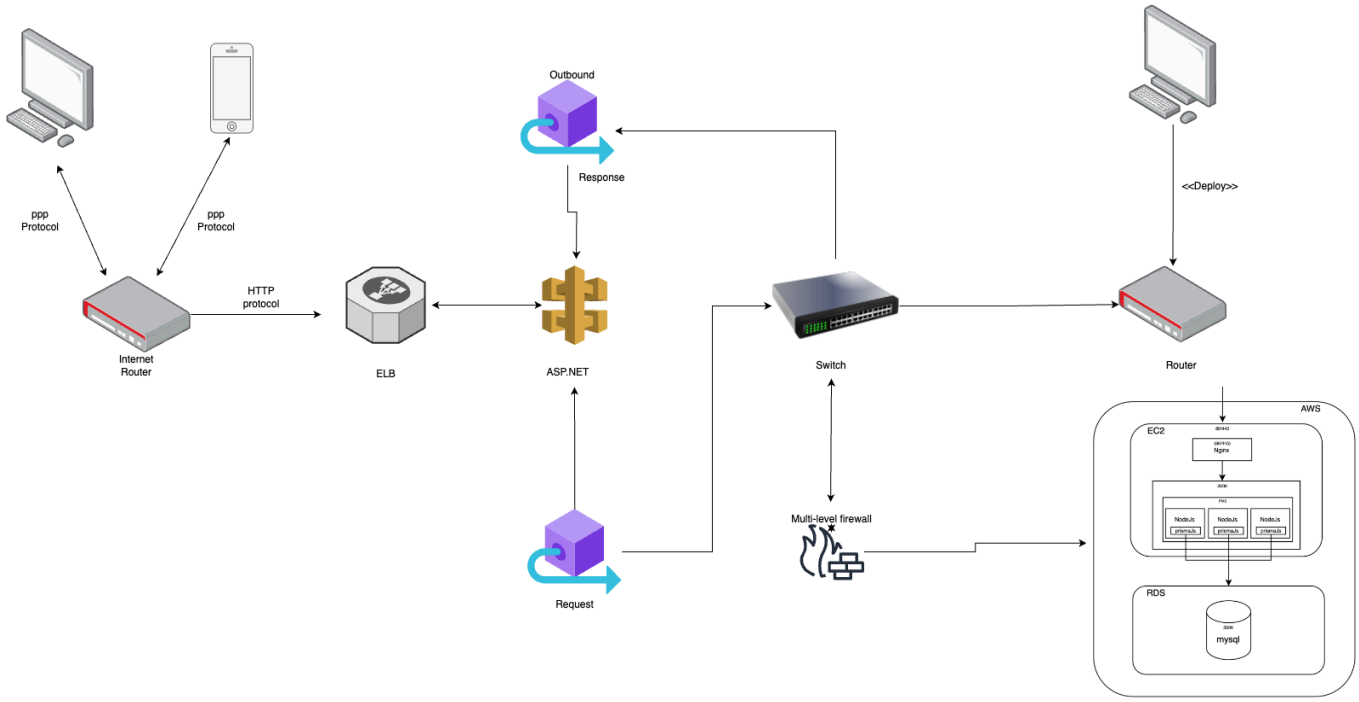
- If the request is for data that doesn't change often, Amazon ElastiCache steps in to provide this data quickly, as it's a temporary storage that remembers frequently accessed information.

- 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 -Registration API -Create Game API -[BE - API] Get User Profile -[BE - API] View Games -UML -Scalability Diagram

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