CPS 731 Final Report

Online Pickup Sports System

(Group 12)

Name	Student Number	Email
Rostyslav Pron	500 909 362	rpron@ryerson.ca
Jenil Vekaria	500 817 039	jvekaria@ryerson.ca
Deep Patel	500 836 204	d14patel@ryerson.ca
Adam Ferrara	500 695 876	adam.ferrara@ryerson.ca
Muhammad Siddiqui	500 837 501	m5siddiqui@ryerson.ca

1 Problem Statement

Sports are enjoyed by people across the world throughout the year. Typically, sports are played in a variety of styles ranging from one-on-one play all the way to unofficial leagues, however getting the whole friend/sport group together all the time can be difficult. Managing the schedule of all the players and/or unforeseen circumstances can be extremely challenging and can result in suboptimal playing conditions.

Your development team is tasked with the responsibility to create an application/service which will allow people near the end-user's location to post requests for extra players in the scenario that they are missing players for the sport in question. The application will also work on a first-come-first-serve basis and will allow the users to join a group that they can play with.

2 Informal Requirements Description

Once the players sign up and log into the application, they will land on the homepage which will have a Google map where they can enter their postal code or enable location to find the nearest sport game that requires extra players. Players can filter sport games on the Google Map by types of sport, skill-level, age (or age group - for privacy), distance, and scheduled date and time. The Google Map will consist of pins that represent all the nearby games and the user can select one to view more information.

On the information page, it will have the game host name, address, scheduled date and time, pictures (optional), description, option to join the team and request for the host's phone number. If the player decides to join the game, the host will be notified with the player's information and can view the list of requests, and decide whether they want to accept the request. In any circumstance where the player decides not to join the game, they can cancel their join request. Cancelling the join request will update the host's game size, and spots will open for any other players who may want to join. Once all the spots have been taken, the posting will disappear and will no longer be visible on the map.

Players who want to host a game and request for extras players can put their posting on the application. Posting will require them to enter the address, scheduled date and time, description, pictures (optional), and game size + [number of players being requested for]. The earliest the request can be posted is 2-3 days prior to the scheduled game time. The host and players will be able to view all the players who have joined the game as well as detailed information of each player.

3 Process Model

The software process model the project will be using is Incremental development. Since the project is not a critical application, this process model best fits the project as it will be delivered incrementally. The project does anticipate requirements changes during the development phase and the incremental process model will incorporate the changes with minimal issues. Since the project is delivered incrementally, it will be easier to test each delivered feature as well as assess how the application is performing in the early stage.

4 Use Case Model

The use case model of the Online Pickup Sports has already been established. The use cases and their relationships are summarized in Figure 1. Detailed information of each use case are given below.

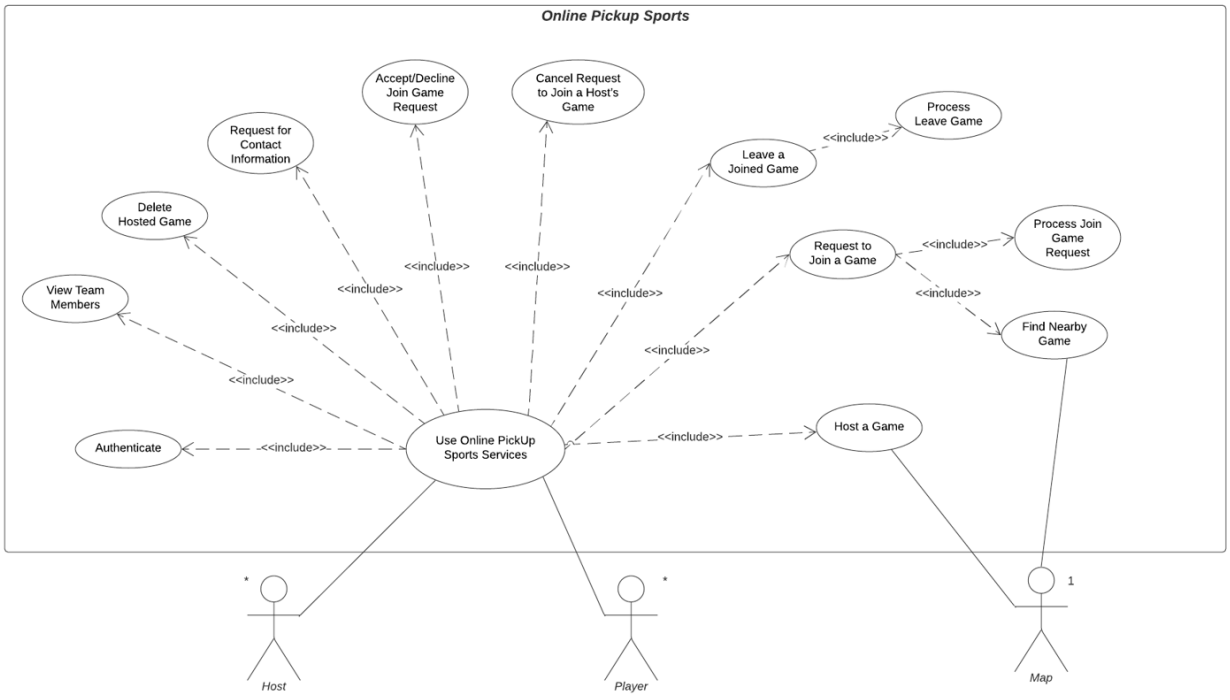


Figure 1: Online Pickup Sports Use Case Diagram

Use Online PickUp Sports Services Use Case

Use Case: Use Online PickUp Sports Services
Scope: Online Pickup Sports
Level: Summary
Intention: The intention of the *User* is to join an existing sport game happening near them using Google map or host a sport game up to a few days in advance. They can also manage their activities.
Multiplicity: Many *Users* can join or host a sport game and manage their activities.
Primary Actor: *User* (can be either *Player* or *Host*)
Main Success Scenario:

1. *User* authenticates into the system

2. *User* proceeds to request to join a game , cancel request to join a host's game , leave a joined game , host a game , accept/decline join game requests , delete hosted game , view team members information , or request for host phone number
3. *User* informs *System* they wish to log out
4. *System* informs *User* of successful log out.

Extensions:

1a. Authentication is not successful. Use case end in failure.
1b. The *User* is already authenticated. Use case continues at step 2.
1c. The *User* does not have an account with the *System*
    1c.1 The *User* informs the *System* that he/she wishes to register, providing registration details.
    1c.2 The *System* confirms registration to the *User*. Use case continue at step 2

Authenticate Use Case

Use Case: Authenticate
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *User* is to authenticate with the *System*
Multiplicity: Multiple *Users* can authenticate simultaneously. A given *User* can only authenticate once at a given time.
Primary Actor: *User* (can either be *Player* or *Host*)
Main Success Scenario:

1. *User* provides username and password to the *System*
2. *System* validates the username and password
3. *System* informs *User* of successful login

Extensions:

2a. *System* informs *User* that the credentials are wrong or unknown
    2a.1. *System* prompts *User* to try again. Use case continues at step 1

Request to Join a Game Use Case

Use Case: Request to join a game
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Player* is to find a game and place a request to join the game
Multiplicity: Many *Players* can join a game
Primary Actor: *Player*
Main Success Scenario:

1. The *Player* uses the Google *Map* to find nearby game by entering their postal code and selecting a distance range from their entered location

At this point the Player can filter the results to only show their favourite sports

2. The *Player* place a join game request for the selected nearby game
3. *System* process join game request

Find Nearby Game

Use Case: Find nearby game
Scope: Online Pickup Sports
Level: Subfunction
Intention: The intention of the *System* is to find all the nearby game given a postal code
Primary Actor:
Secondary Actor: *Map*
Main Success Scenario:
1. The *System* provides the *Map* the postal code
2. The *Map* provides the *System* with all the nearby game
   At this point, all the games have open spots and are available for joining

Extensions:
1a. The *System* provides an invalid postal code to the *Map.* Use case ends in failure.
2a. The *Map* is unable to find any nearby game for the given postal code. Use case ends in failure.

Process Join Game Request Use Case

Use Case: Process Join Game Request
Scope: Online Pickup Sports
Level: Subfunction
Intention: The *System* needs to check if the request to join a game is valid
Multiplicity: Multiple requests might have to be processed concurrently
Primary Actor:
Secondary Actor: *Player, Host*
Main Success Scenario:
1. The *System* checks if there are enough player spots for the *Player* to join the game
2. The *System* ensures that *Player* is not part of another game for the same time slot
3. The *System* informs both the *Player* and *Host* of successful request

Extensions:
1a. System informs *User* that the game has reached full capacity. Use case ends in failure.
2a. *System* informs *User* is already part of another game for the same time slot. Use case ends in failure.

Cancel Request to Join a Host's Game Use Case

Use Case: Cancel request to join a host's game
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Player* is to cancel a join game request.
Multiplicity: Many *Players* can cancel many join game requests at any given time.
Primary Actor: *Player*

Main Success Scenario:

1. The *Player* informs the *System* he wants to cancel the join game request
2. The *System* acknowledges the cancellation to the *Player* and informs *Host* that the *Player* has canceled

Extensions:

2a. If the game has already started with a pending request, the *System* informs *Player* the game has started and cancels the pending request successfully.

2a. The *Player* cancel a request for a game that is in-progress

2a.1 The *System* informs *Player* the pending request has been cancelled successfully

Leave a Joined Game Use Case

Use Case: Leave a joined game
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Player* is to leave a joined game.
Multiplicity: Many *Players* can leave many joined games.
Primary Actor: *Player*
Secondary Actor: *Host*
Main Success Scenario:

1. The *Player* informs the *System* he wants to leave the game
2. The *System* process leave game

Process Leave Game Use Case

Use Case: Process leave game
Scope: Online Pickup Sports
Level: Subfunction
Intention: The *System* needs to remove the *Player* from the game.
Multiplicity: Multiple cancellation might have to be processed concurrently
Primary Actor:
Secondary Actor: *Player, Host*
Main Success Scenario:

1. The *System* removes *Player* from the game and open a new spot any potential new *Players*
2. The *System* ensures that the game is visible on the Map and available for new *Players* to join
3. The *System* informs *Player* he has left the game successfully and *Host* is notified

Extensions:

2a. The game reaches capacity and the *System* removes it from the Map

Host a Game Use Case

Use Case: Host a game
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention is for the *Host* to host a game so that any nearby players can join the game.

Multiplicity: Many *Hosts* can host as many games they like. One *Host* can only host one game at one specific time.
Primary Actor: *Host*
Secondary Actor: *Map*
Main Success Scenario:
1. The *Host* informs the *System* to create a game
2. The *Host* provides all the necessary game information
3. The *System* informs the *Host* that the game has been created successfully

Extensions:
3a. The *System* informs the *Host* that the game creation was unsuccessful.
3a.1 The *Host* creates a game with the date/time same as a pre-existing game. Use case ends in failure.
3a.2 The *Host* provides a date/time that is in the past or too far in the future. Step 2 is redone

Accept/Decline Join Game Request Use Case

Use Case: Accept/Decline Join Game Request
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Host* is to view all the join game requests and decide whether they want to accept or decline the request.
Multiplicity: Many *Hosts* can accept or decline all the join game requests for their respective hosted game.
Primary Actor: *Host*
Secondary Actor: *Player*
Main Success Scenario:
1. The *Host* informs the *System* he/she wishes to review the join game requests.
2. The *System* acknowledges the *Host* that a request has been accepted or declined successfully.
3. The *System* informs the *Player* of their join game request status.

Extensions:
2a. The *Player* cancels their join game request and is no longer available in the *Host*'s request queue. Use case ends in failure.

Delete Hosted Game Use Case

Use Case: Delete Hosted Game
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Host* is to delete a hosted game they no longer.
Multiplicity: Many *Hosts* can delete many of their hosted games at any time
Primary Actor: *Host*
Secondary Actor: *Map*
Main Success Scenario:
1. The *Host* informs the *System* he/her wants to delete the hosted game
2. The *System* deletes the game and removes its from the *Map*
3. The *System* informs the *Host* and all the *Players* that the game has been deleted

Request for Contact Information Use Case

Use Case: Request for contact information
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *Host* is to gain a means of contacting *Players* (who are requesting to join) and *vice versa* for ease of management and organization (e.g. games that need equipment, etc.).
Multiplicity: One *Host* can ask many *Players* (requesting to join their game) for their contact information. One *Player* can request from one *Host* for their contact information.
Primary Actor: *User* (either *Player* or *Host*)
Main Success Scenario:
1. The *Host* informs the *System* to view the list of *Players* requesting to join the team
2. The *Host* informs the *System* to request a *Player*'s contact information
3. The *System* informs the *Player* of a request from the *Host* to view their contact information
4. The *Player* accepts the request
5. The *System* informs the *Host* of the *Player*'s contact information

Extensions:
1a. The *Player* informs the *System* to view a specific game's details to see the *Host*
    1a.1 The *Player* informs the *System* to request the *Host*'s contact information
    1a.2 The *Host* accepts the request
        1a.2a The *Host* declines the request to share their contact information. Use case ends in failure.
    1a.3 The *System* informs the *Player* of the *Host*'s contact information
4a. The *Player* declines the request to share their contact information. Use case ends in failure.

View Team Members Use Case

Use Case: View Team Members
Scope: Online Pickup Sports
Level: User-goal
Intention: The intention of the *User* is to view all other *User*'s profiles who have been accepted to the game
Multiplicity: Many *Users* are able to view many *User's* profiles in the same team
Primary Actor: *User*
Main Success Scenario:
1. The *User* informs the *System* to view details of the specific game they are a part of
2. The *System* displays to the *User* a list of all other *Users*' profiles (*Host* and *Players* accepted into the game)

Extensions:
2.1 The *User* selects a *User*'s profile
2.2 The *System* displays the selected *User*'s profile summary, including contact information

5 Non Functional Requirements

Reliability
The System shall ensure that the GPS and Map provides accurate locations to the user to ensure the user is not misled in any way. The number of errors for locations should be minimal, not exceeding more than 2 per hour of system use.

Availability
The System is a fully online service, entirely dependent on having a wifi connection and access to GPS or location; it should be available to users during (expected) peak usage hours (Mon-Fri, 12pm-6pm). Downtime during peak hours shall not exceed more than one minute..

Timeliness
The System shall ensure the timeliness of updates to games, and updates to the Map so that the user will have accurate information available at all times for decision making (e.g. choosing what game to join, whether they want to cancel, etc.). It is anticipated that users of the system will make many rapid changes from short notice, so in order to let the relevant users be notified in a timely manner, the system should be able to respond to requests within 3 seconds.

Robustness
The System should be robust to user actions that can lead to errors, and provide an appropriate and understandable message to users should their action be invalid (e.g. invalid inputs: if a game is deleted the moment a user requests to join it). The percentage of events causing failures should be no more than 10%.

Confidentiality
The System shall not disclose any user's personal information (profile and contact information) to other users without their informed consent, i.e. when another user specifically asks for contact information, or when a user is requesting to join a game (thereby offering their contact information for communication thereon). Otherwise, a user's information should be able to be kept private to all other users until made public by the respective user.

Organization
Users of the system shall authenticate themselves using their email address.

6 Domain Model

Figure 2 shows the domain model that has been established during requirement elicitation to accompany the use case model.

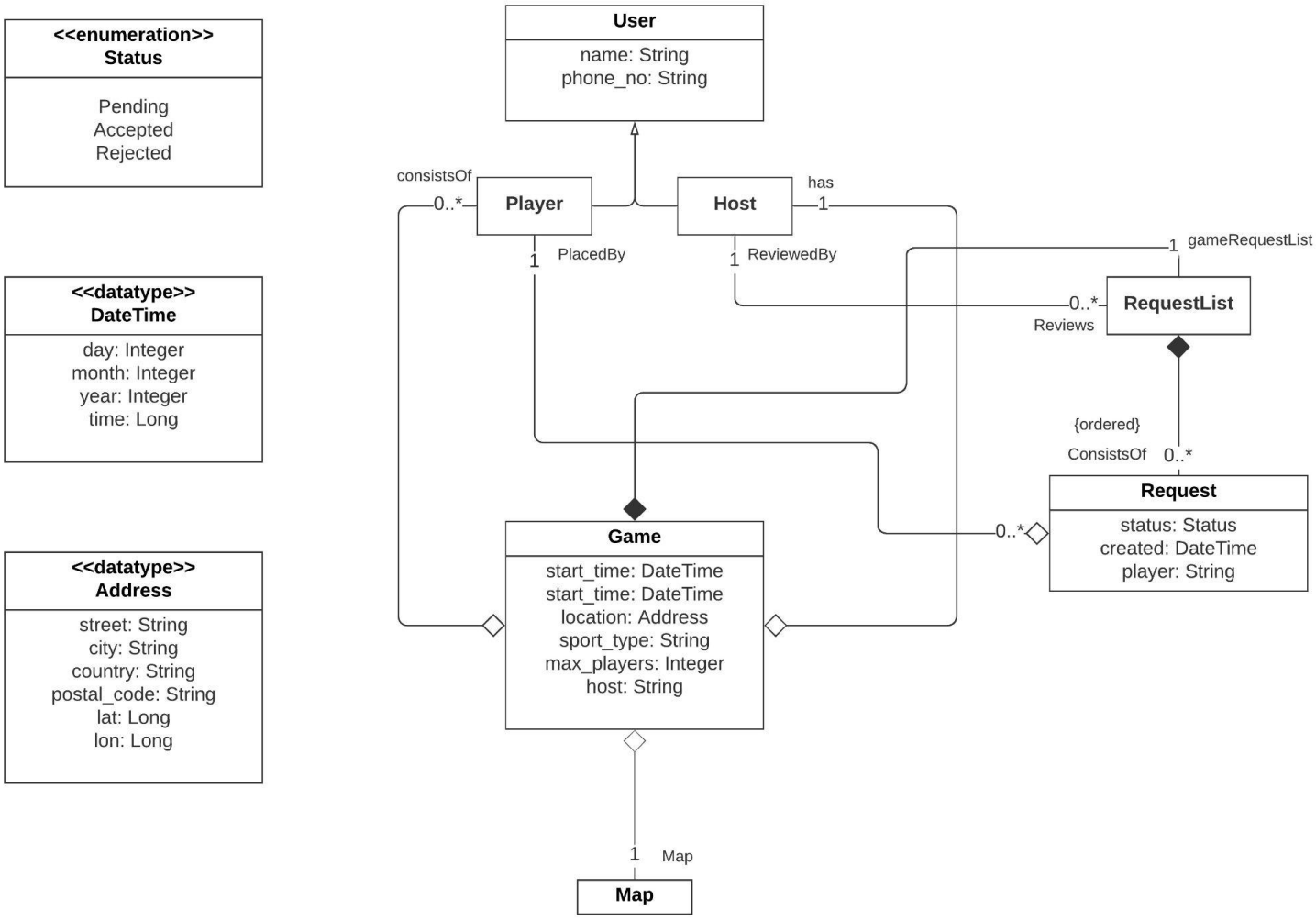
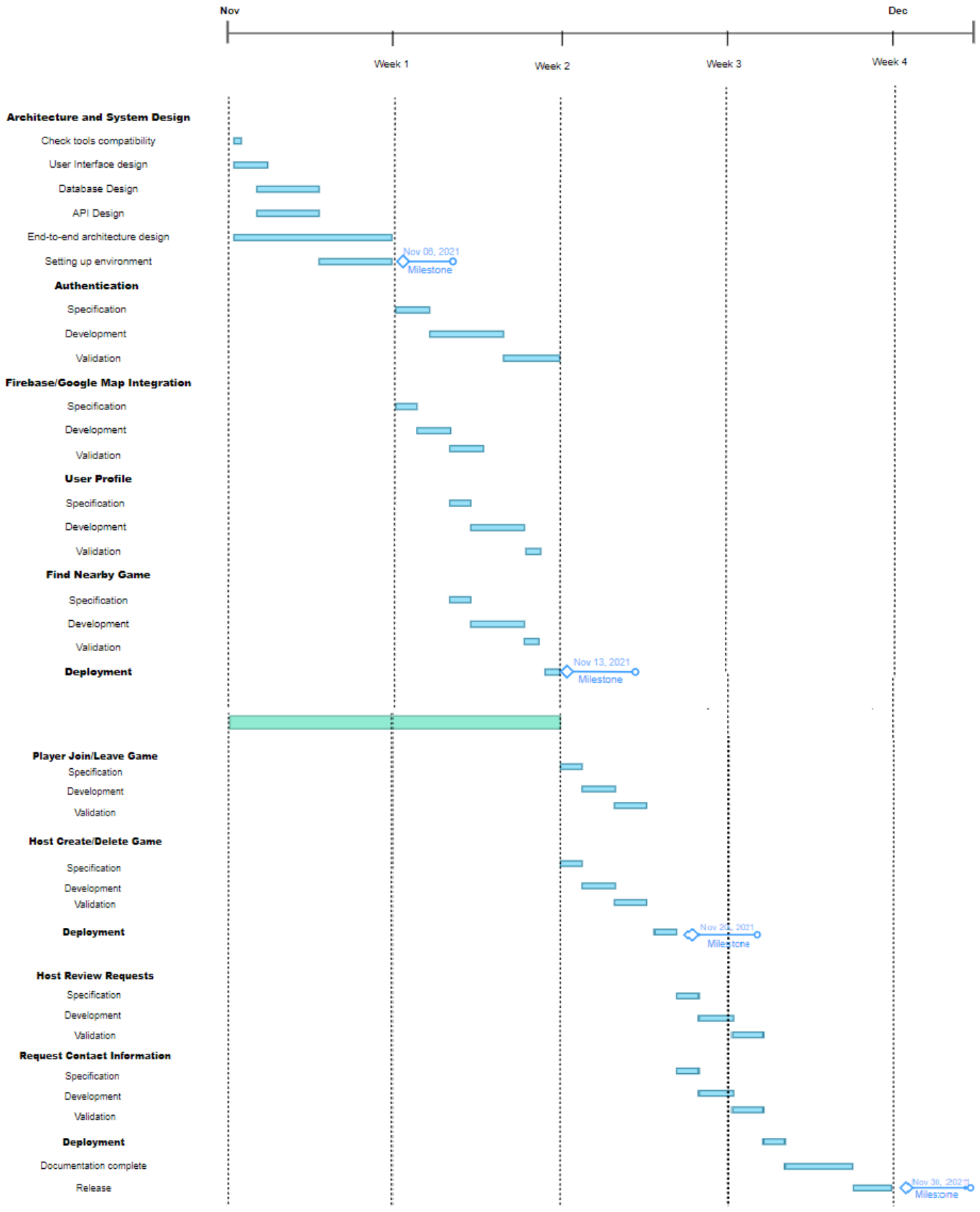


Figure 2. Online Pickup Sports Domain Model

7 Project Plan



We have established a time estimate for each task below.

Activity	Time Estimate (in Days)
1. Architecture and System Design	
1.1 Check Tools Compatibility	1
1.2 User Interface Design	2
1.3 Database Design	3
1.4 API Design	3
1.5 End-to-End Architecture Design	7
1.6 Setting up environment	4
2. Authentication	
2.1 Specification	2
2.2 Development	3
2.3 Validation	2
3. Firebase/Google Map Integration	
3.1 Specification	1
3.2 Development	2
3.3 Validation	2
4. User Profile	
4.1 Specification	1
4.2 Development	3
4.3 Validation	1

5. Find Nearby Game	
5.1 Specification	1
5.2 Development	3
5.3 Validation	1
6. Deployment	
6.1 Deployment	1
7. Player Join/Leave Game	
7.1 Specification	1
7.2 Development	2
7.3 Validation	2
8. Host Create/Delete Game	
8.1 Specification	1
8.2 Development	2
8.3 Validation	2
9. Deployment	
9.1 Deployment	1
10. Host Review Requests	
10.1 Specification	1
10.2 Development	2
10.3 Validation	2

11. Request Contact Information	
11.1 Specification	1
11.2 Development	2
11.3 Validation	2
12. Deployment	
12.1 Deployment	1
13. Production Release	
13.1 Documentation complete	5
13.2 Release	3

Below we have established the slack time for all the activities.

Activity	Earliest Start Date	Early Finish	Latest Start Date	Slack
1.1	0	1	1	1
1.2	0	2	0	0
1.3	2	5	2	0
1.4	5	8	5	0
1.5	8	15	8	0
1.6	15	19	15	0
2.1	19	21	19	0
2.2	21	24	21	0
2.3	24	26	24	0
3.1	19	20	21	2
3.2	20	22	22	2

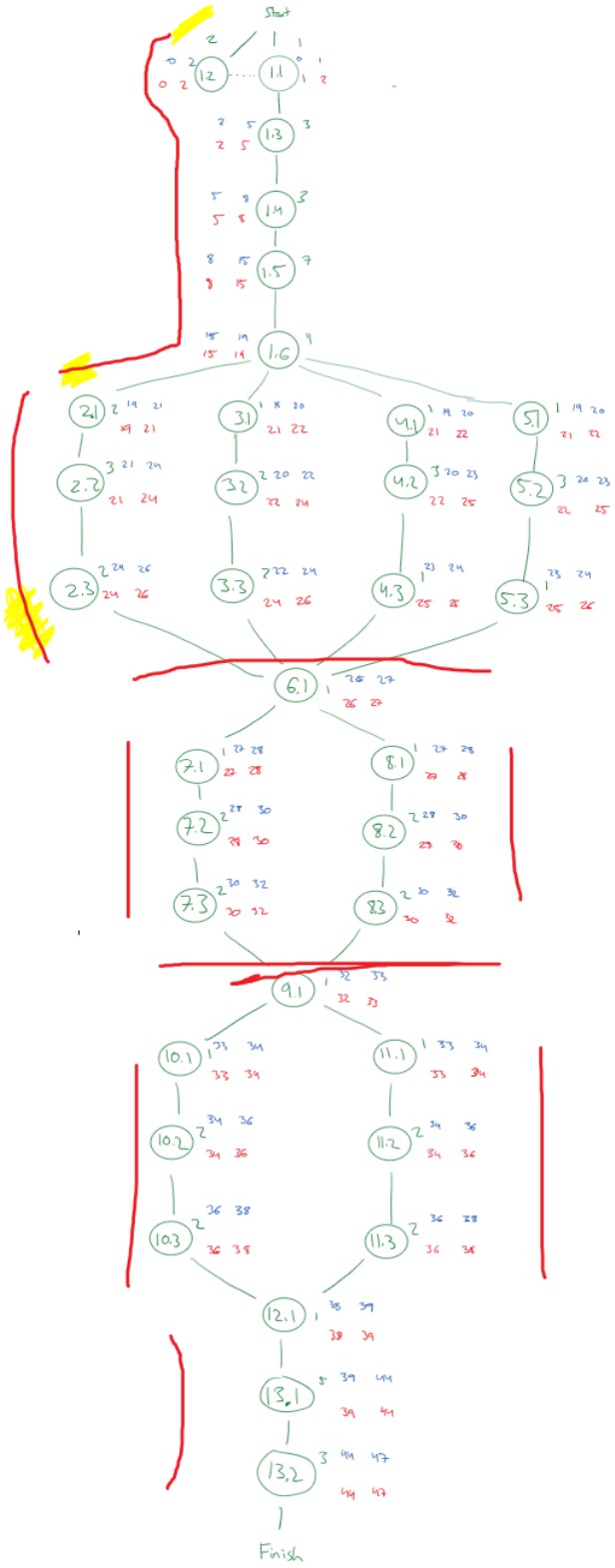
3.3	22	24	24	2
4.1	19	20	21	2
4.2	20	23	22	2
4.3	23	24	25	2
5.1	19	20	21	2
5.2	20	23	22	2
5.3	23	24	25	2
6.1	26	27	26	0
7.1	27	28	27	0
7.2	28	30	28	0
7.3	30	32	30	0
8.1	27	28	27	0
8.2	28	30	28	0
8.3	30	32	30	0
9.1	32	33	32	0
10.1	33	34	33	0
10.2	34	36	34	0
10.3	36	38	36	0
11.1	33	34	33	0
11.2	34	36	34	0
11.3	36	38	36	0
12.1	38	39	38	0
13.1	39	44	39	0
13.2	44	47	44	0

The following series of activities are the critical path.

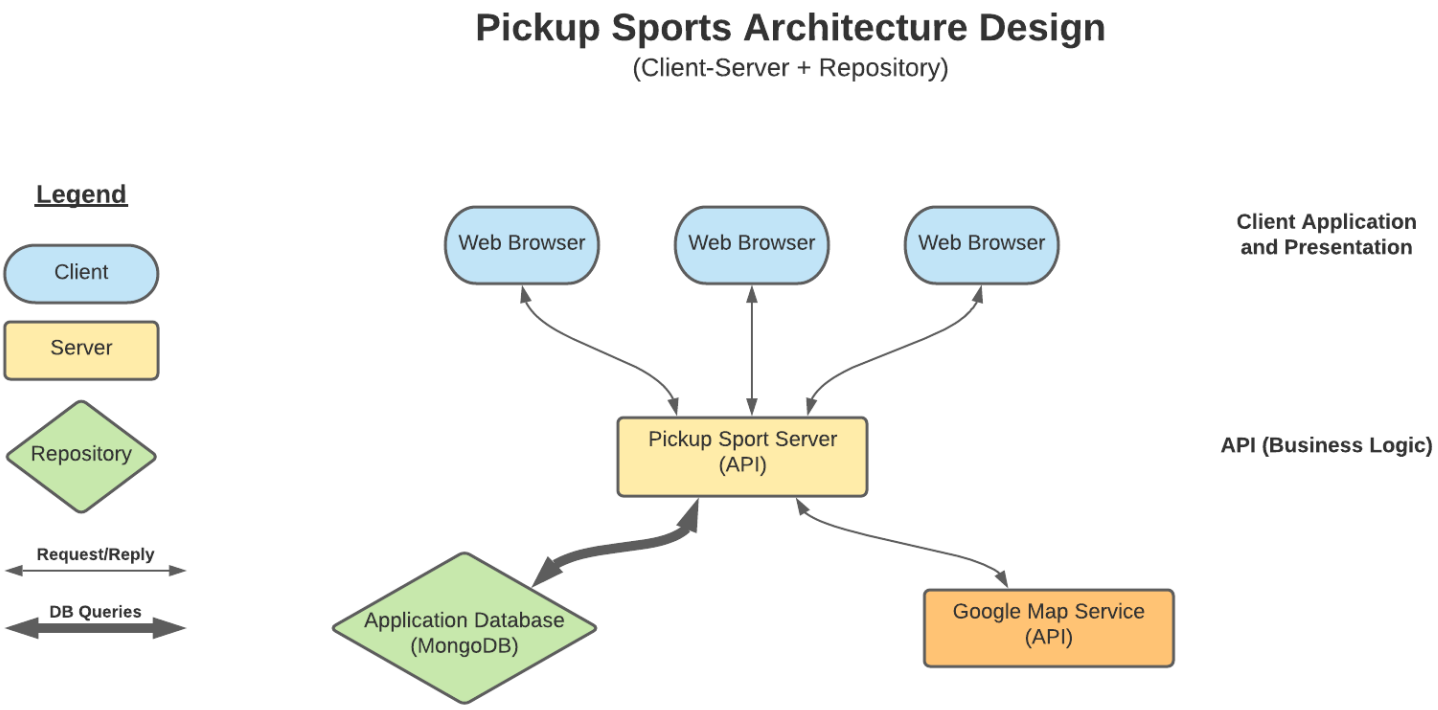
1.2 - 1.6
2.1 - 2.3
6.1 - 13.2

Note: from 6.1 there are 2 concurrent branches (7.1-7.3, and 8.1-8.3). Likewise, from 9.1 there are also 2 concurrent branches (10.1-10.3, and 11.1-11.3) which are all critical paths.

Activity Diagram, (The red markings denote the critical path, the annotations are for convenience in calculating slack (early start, early finish, etc.)). Activity Diagram (for reference)

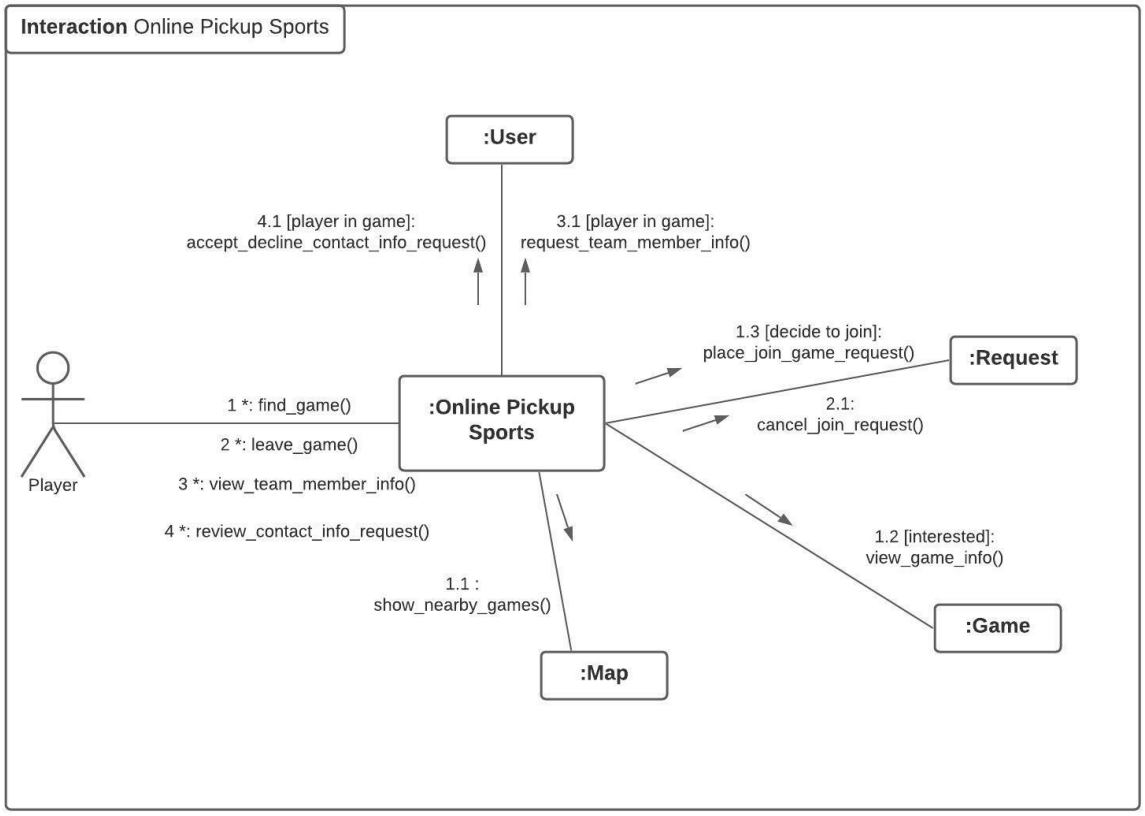


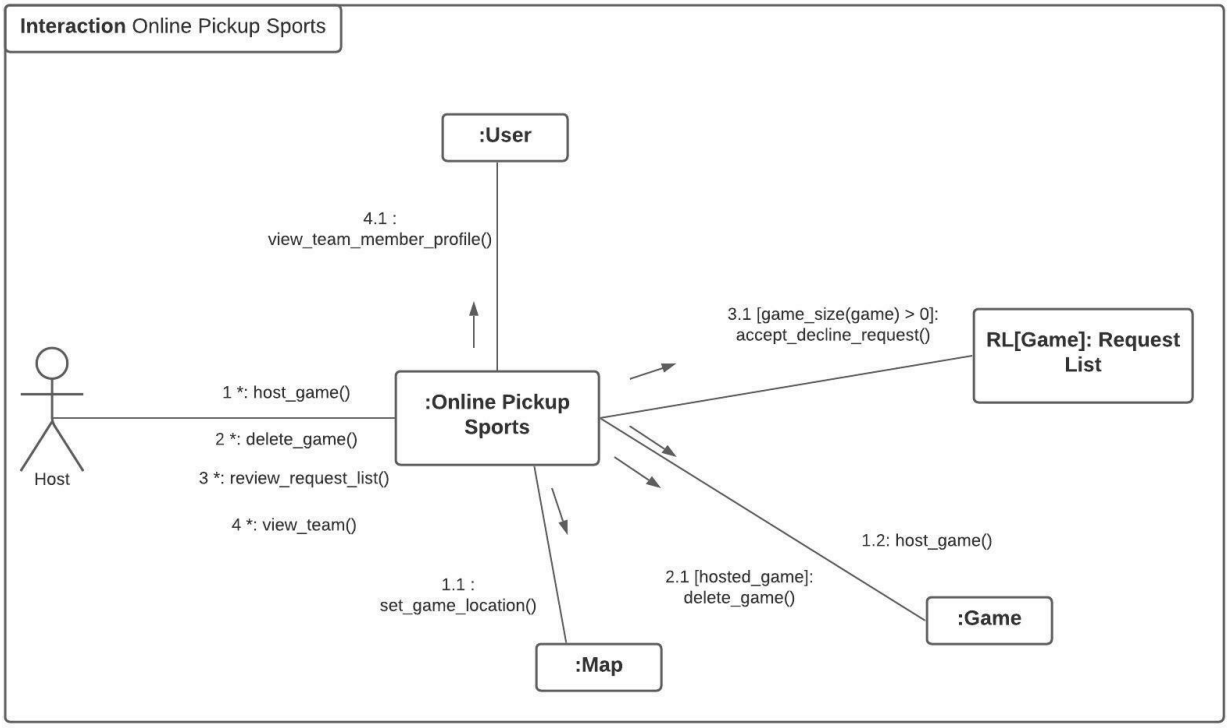
8 Architectural Design



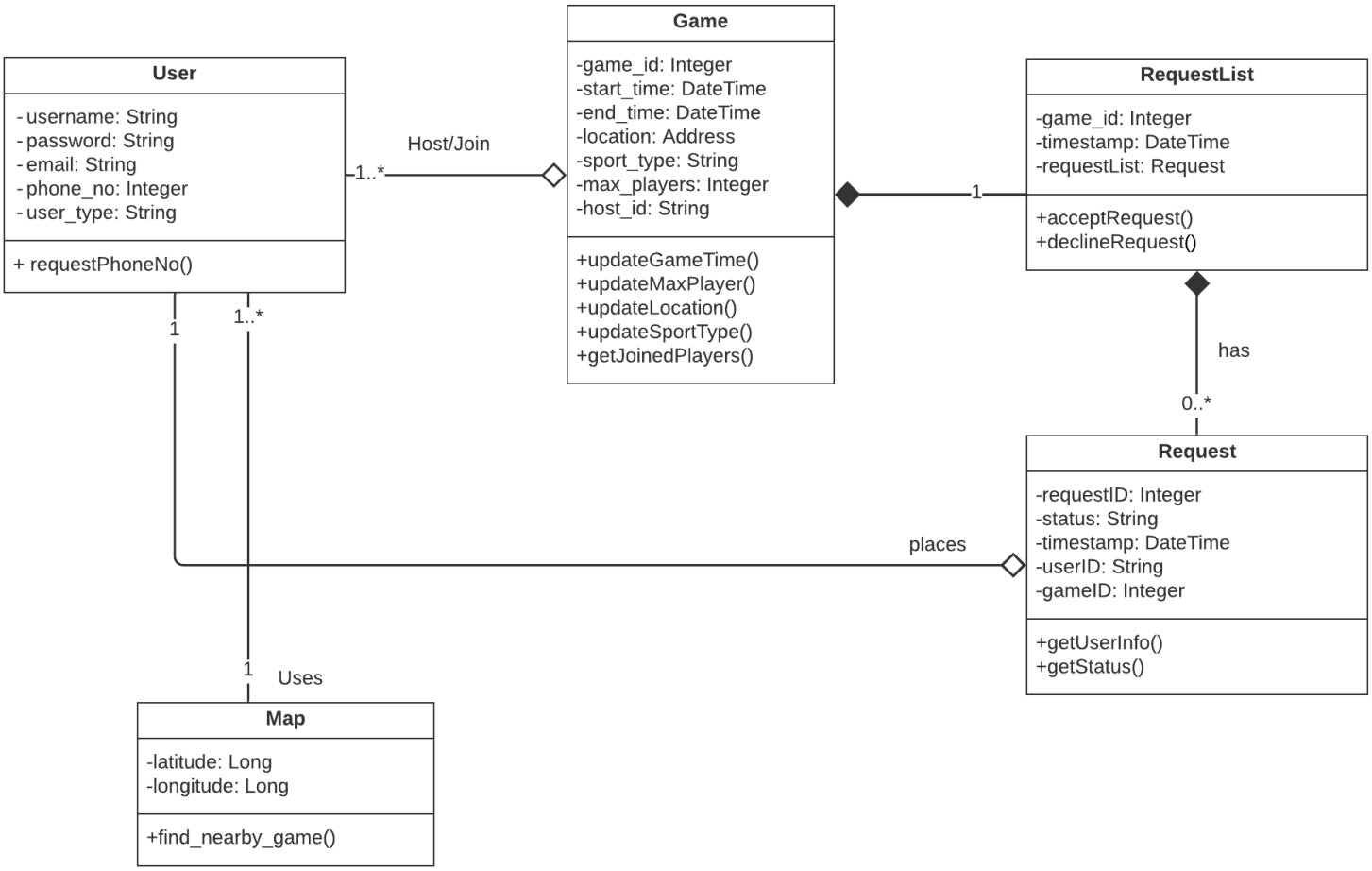
Our group decided to use a combination of Client-Server and Repository architecture. This architecture is appropriate for our system because the player/host will need to interact with our server (API) which will handle operations such as host/delete game, place/review game request, find nearby game, authentication, view team members and leave game. These operations on the server will directly work with our repository (MongoDB) to do read/write operations. The server will also communicate with the Google Map Service API to find nearby game with the filter (eg. sports type, distance range, time) user has provided.

9 Interaction Model - Communication diagram

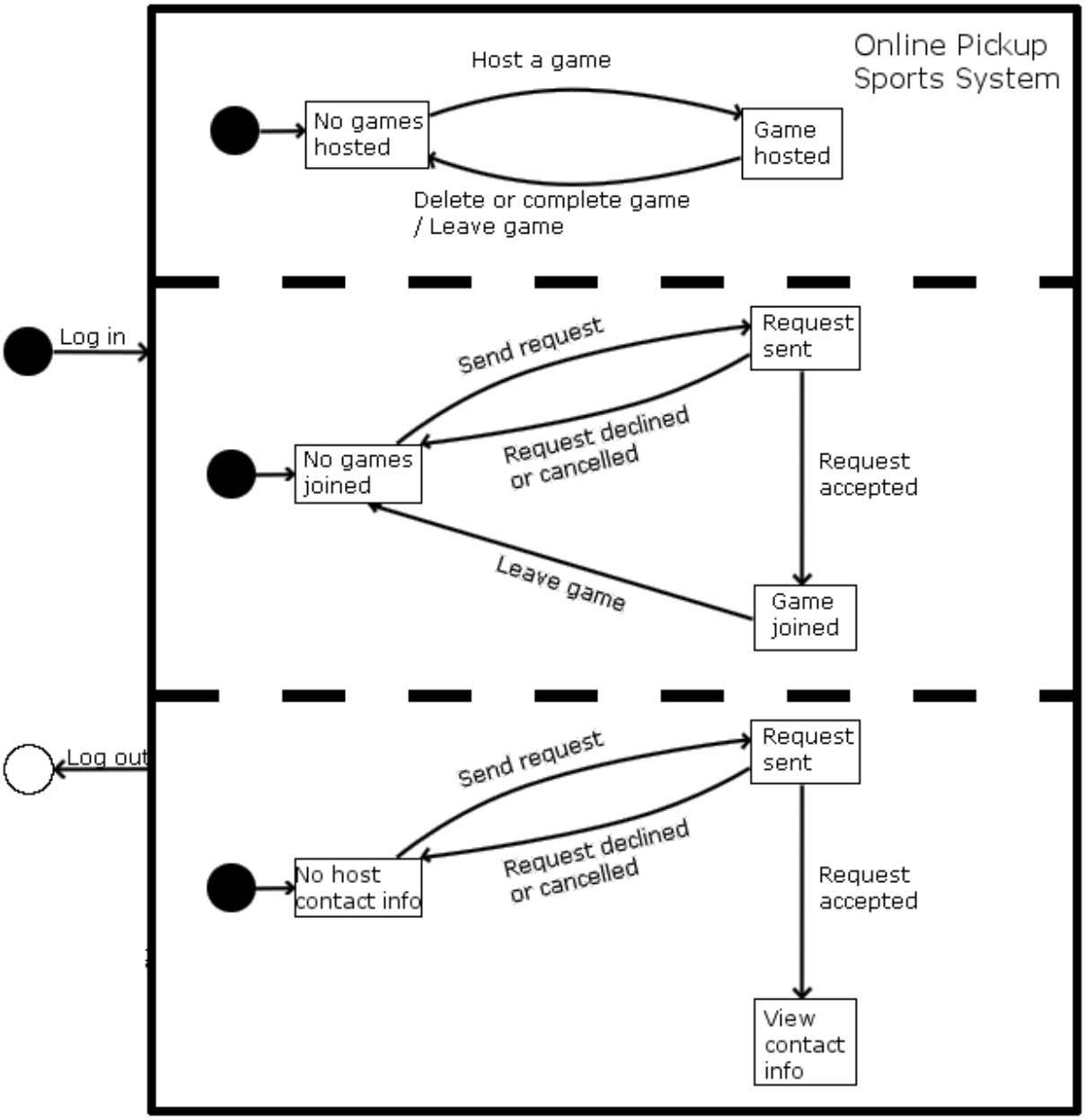




10 Design Class Model



11 State Machine Model



12 Test Design

Authentication

Test Cases	Description	Input	Expected Result
Task 1: Authentication			
1A	User provides username and password to the system	Username & Password	User should be able enter their credentials
1B	System validates the username and password	Username & Password	System should be able to validate the user credentials
1C	System informs user of successful login	N/A	View success message
Task 2: Authentication (Invalid input)			
2A	User provides username and password to the system	Username & Password	User should be able to enter their wrong credentials
2B	System informs user of unsuccessful login	N/A	View error message
2C	System ask user to re-enter the username and password	N/A	Show message to re-enter username and password

Accept/Decline Join Game Request

Test Cases	Description	Input	Expected Result
Test Case 1: Accept/Decline Request			
1A	Host accept/decline join game request	Press accept or decline button	Host should be able to press the buttons
1B	System processes the accept/decline game request	N/A	System notifies the player of their request status and host should be able to success message

Request to Join a Game

Test Cases	Description	Input	Expected Result
Test Case 1: Request to join a game			
1A	User places a join game request	Press join game request button	Join game request is sent to system
1B	System processing the game request	N/A	System acknowledges request and sends success criteria
1C	Request is approved by system	N/A	System shows success message

Find Nearby Game

Test Cases	Description	Input	Expected Result
Test Case 1: Find nearby game			
1A	User provides postal code/distance range to find nearby games	Postal Code and distance	User should be able postal code and distance range
1B	System finds nearby game	N/A	System shows list of nearby game
Test Case 2: Find nearby game (Invalid input)			
2A	User provides postal code and distance to find nearby games	Postal Code and distance	User should be able to enter an invalid postal code
2B	System informs user of invalid input	N/A	User sees error message
2C	System ask user to re-enter a valid postal code	N/A	Show message to re-enter
Test Case 3: No Nearby Game			
3A	User provides postal code and distance to find nearby games	Postal Code and distance	User should be able postal code and distance range
3B	System cannot find find any nearby game	N/A	System shows message that there are no nearby games

Process Join Request

Test Cases	Description	Input	Expected Result
Test Case 1: Process Request			
1A	User places a join game request	Press join game request button	Join game request is sent to system
1B	Request is approved by system	N/A	System shows success message
Test Case 2: Cannot Process Request			
1A	System processes the game request	Game request details	System shows processing message
2B	System cannot process request since game reached full capacity	N/A	System shows failure message

Cancel Join Game Request

Test Cases	Description	Input	Expected Result
Test Case 1: Cancel Join request			
1A	User cancels a join game request	Press cancel game request button	Cancel game request is sent to system
1B	System process the cancel join game request	N/A	System shows success message and game slot is open
Test Case 2: Cancel Pending request (Game Started)			
2A	User cancels a join game request	Press cancel game request button	Cancel game request is sent to system
2B	System determines game has started and cancels the join request	N/A	System shows success message
Test Case 3: Cancel Pending request (In-progress Game)			
3A	User cancels a join game request	Press cancel game request button	Cancel game request is sent to system
3B	Game is in progress and join request is cancelled	N/A	System shows success message

Leave a Joined Game

Test Cases	Description	Input	Expected Result
Test Case 1: User leave joined game			
1A	User leaves joined game which hasn't started yet	Press leave game button	User is able to press leave button
1B	System removes user from the game	N/A	System shows leave game success message

Process Leave Game

Test Cases	Description	Input	Expected Result
Test Case 1: Process leave game			
1A	System removes user from the game	N/A	System should be able to open extra game slot and game should be visible on the map

Delete Hosted Game

Test Cases	Description	Input	Expected Result
Test Case 1: Delete Game			
1A	Host deletes a game	Press delete game button	Host should be able to press cancel game button
1B	System processes the game deletion	N/A	System notifies the player that the game has been cancelled and game should no longer be available on the map

Host a Game

Test Cases	Description	Input	Expected Result
Test Case 1: Host a Game			
1A	User decides to host a new game	Press ‘Host a Game’ button	User is able click the ‘Host a Game’ button
1B	User provides game information	Address Sport Type Game Capacity Start Time End Time	User should be able to input valid information for the game
1C	System creates the game	N/A	System creates the game and shows successful message
Test Case 2: Host a Game (Same date/time as existing game)			
2A	User decides to host a new game	Press ‘Host a Game’ button	User is able click the ‘Host a Game’ button
2B	User provides game information	Address Sport Type Game Capacity Start Time End Time	User should be able to input date/time as already existing game
2C	System doesn't creates the game	N/A	System doesn’t create the game and shows error message
Test Case 3: Host a Game (Game date/time is in the past or too far into the future)			
3A	User decides to host a new game	Press ‘Host a Game’ button	User is able click the ‘Host a Game’ button
3B	User provides game information	Address Sport Type Game Capacity Start Time End Time	User should be able to input date/time in the past or way into the future
3C	System doesn't creates the game	N/A	System doesn’t create the game and shows error message

Request for Contact Information

Test Cases	Description	Input	Expected Result
Test Case 1: Request Player Information (Accepted Request)			
1A	Host decides to view list of player requests	Press ‘View Requests’ button	User is able click the ‘View Requests’ button
1B	Host selects a player request and asks for additional information	Press ‘Request Player Information’ button	User is able click the ‘Request Player Information’ button
1C	System informs Player of request to view additional information	N/A	System shows Player a prompt stating that the Host is requiring additional information about the player
1D	Player accepts request	Press ‘Accept’ button	Player should be able to press the ‘Accept’ button
1E	System shows the Host the request players information	N/A	System shows player information to the host
Test Case 2: Request Player Information (Rejected Request)			
2A	Host decides to view list of player requests	Press ‘View Requests’ button	User is able click the ‘View Requests’ button
2B	Host selects a player request and asks for additional information	Press ‘Request Player Information’ button	User is able click the ‘Request Player Information’ button
2C	System informs Player of request to view additional information	N/A	System shows Player a prompt stating that the Host is requiring additional information about the player
2D	Player declines request	Press Decline button	Player should be able to press the ‘Decline’ button
2E	System informs Host that player declined their request	N/A	System informs the Host that the player has declined their request
Test Case 3: Request Host Information (Success)			
3A	Player decides to view additional information about host	Press ‘Request Host Information’ button	User is able click the ‘Request Host Information’ button

3B	System informs Host of request to view additional information	N/A	System shows the Host a prompt stating that a Player is requiring additional information about the Game/Host
3C	Host Approves request	Press Approve button	Player should be able to press the ‘Approve’ button
3D	System shows the Player the requested Hosts’ information	N/A	System shows Host information to the Player
Test Case 4: Request Host Information (Failure)			
4A	Player decides to view additional information about host	Press ‘Request Host Information’ button	User is able click the ‘Request Host Information’ button
4B	System informs Host of request to view additional information	N/A	System shows the Host a prompt stating that a Player is requiring additional information about the Game/Host
4C	Host Declines request	Press Declines button	Player should be able to press the ‘Decline’ button
4D	System informs Player that the Host has declined their request	N/A	System informs the Player that the Host has declined their request

View Game Team Members Info

Test Cases	**Description**	**Input**	**Expected Result**
Test Case 1: View team member info			
1A	User viewer all the user’s profile who have joined the game	Press view profile	User should be able to see the other user’s profile

13 Implementation and Testing

By the beginning of December, the website system was largely feature-complete and running successfully using React as a front-end and Express js as a back-end. Tests were successfully carried out by downloading copies of the repository, running the downloaded copies on client machines, and making changes and edits whenever necessary. Through consistent testing, all necessary features were able to be completed to a satisfactory point, including the Google Maps integration, the log-in and authentication system, the search function, the request-sending and joining functions, and the hosting system.

14 Project Plan Review

The software process model that was proposed in the first phase report is Incremental development. Generally speaking, the team was able to adhere to this model throughout the second phase, though some minor difficulties arose around the end due to some factors outside of the team's control, such as conflicting schedules and issues connecting with the VPN. Despite these, the team as a whole was able to use incremental strategy successfully to first present mockups of the intended designs, implement them in smaller-scale tests to verify their functionality, then finally implement these increments successfully into the main build. These practices were held for both important components such as the request, map, and join features. This was also true for smaller components of the system, such as UI cards representing games, functionality for a "light theme" and "dark theme" that could be switched on a button press, etc.

Project Feature Changes:

There were certain changes to the initial requirements, and one requirement was scrapped in development. Most notably, the "Request for Contact Information" use case and functionality was not implemented due to time constraints, as it was viewed as less important than the other functions of the system. The team also came to the conclusion that the system would simply be more intuitive if it displayed contact information from the UI directly. The use case for searching by postal code was also changed with a simpler Google Maps area search for a better user experience for game search results.

As overall team leadership was largely handled by Jenil and Deep, they were primarily responsible for both setting up increment time periods and ensuring that said increments were completed to the extent specified within the given time. Given time was based around each team member's free time within their schedules to ensure that no excessively large tasks were given to team members whose schedules could not accommodate them.

15 Final Conclusion

Communication among the team was overall consistent. All members generally pulled their weight in regards to performing necessary tasks, though in the end, some members did end up completing slightly more than others. While each team member provided code for the system, each member handled additional duties as well. Jenil and Deep were largely responsible for overall team leadership, including scheduling and holding team meetings to aid in both dividing team work and ensuring the completion of increments. Muhammad provided additional code for the backend and frontend, with Adam and Rostyslav refining the UI as well as organizing, adding to, and proofreading the reports and general system text. Additional difficulties arose in

properly committing changes to the group Git due to issues in properly connecting with the VPN, seemingly both client-side and server-side at times. Due to these issues, certain commits additionally needed to be performed by other group members that retained access to the VPN.

Due to these OpenVPN issues which many students experienced Friday evening to Friday night, the Ryerson servers were inaccessible to students. As a result, some commits and features implemented by students were not able to be updated and merged into the master branch—instead, the files were copy-pasted into the student with the most up-to-date branch in order to continue work.

Note (GitLab commit contributions):

Muhammad Siddiqui (5 commits not shown in GitLab): The commits found in the branch: "Feature/game detail section" (https://gitlab.scs.ryerson.ca/JenilV/pickupsports/-/merge_requests/13) were not able to be merged into the master branch when the Ryerson GitLab server was inaccessible. The code was shared to Jenil Vekaria and is included in this merge into master: https://gitlab.scs.ryerson.ca/JenilV/pickupsports/-/commit/6d10cae90b009d65df6f31d3d0c431a0153f6e39