Gary is Win

Design Use Cases

Team Keep-It-110!

Alex Yu - Project Manager
Angie Thai - Business Analyst
Annie Wong - Senior System Analyst
Davin Tjong - Database Specialist
Katherine Lim - User Interface Specialist
Kim Pham - Quality Assurance Lead
Pasha Bouzarjomehri - Software Development Lead
Rachel Lim - User Interface Specialist
Rama Del Maestro - Algorithm Specialist
Sarah Jung - Software Architect

TABLE OF CONTENTS

Priority (PR):

- **H = high** **(Must have)**
- **M = medium** **(Should have)**
- **L = low** **(Could have)**
- **W = lowest** **(Would be nice to have)**

Status (S):

- **C = Completed**
- **IP = In Progress**
- **N = Not Started**

2.4	Unblock a Player	W	C
2.5	Challenge a Friend to a Match	H	C

3. Game Mechanics			
3.1	Move Avatar	H	C
3.2	Place Shape	H	C
3.3	Win Match	H	C

4. Match Features			
4.1	Play Locally	L	C
4.2	Play Random Opponent	L	C
4.3	Analyze Aggregate Match Statistics	W	N
4.4	Rematch After a Match Ends	L	C
4.5	Prematurely End a Match	M	C
4.6	Chat With Opponent	W	N
4.7	Watch a Replay of a Match	W	N
4.8	Review Moves Made During Match	W	N

5. In-app Features			
5.1	Change Player Avatar	L	C
5.2	Change App Icon	W	C
5.3	Play Tutorial	W	C
5.4	View Leaderboard	L	C
5.5	View Rating	L	C

GLOSSARY OF TERMS

Account	An arrangement by which a user is given personalized access to the game application, by entering a username, email, and password.
Avatar	The 2D figure(s) that represents the player during a match.
Challenge	A method to initiate gameplay, in which a player can start a match and invite another user to play with them.
Guest	A user who is playing the game without an account.
Leaderboard	A scoreboard displaying the rating, names and current scores of the leading competitors in the game.
Match	A single round of gameplay resulting in one player winning and the other losing.
Matchmaking	The process of locating and connecting suitable players together to take part in a multiplayer game session.
Opponent	The player who is playing against a respective player.
Player	A user of the application who is in a match.
Statistics	Data collected and analyzed from completed matches. Includes but is not limited to: win/lose rate, number of moves made, what moves were made.
Turn / Move	A player's turn consists of placing an object tile followed by moving their player tile up to three times.

DESIGN USE-CASE 1.1: Sign-Up

Priority	High - Must have feature
Status	Completed
Description	The user creates a new account by submitting a unique email address, unique username, and password.
User Goal	The user wants to create a new account.
Desired Outcome	The user will have a new account.
Actor	The user of the application.
Dependent Use Cases	N/A
Requirements	SR 1.1
Pre-conditions	There is no existing account associated with the new user's email address or username. If the app was previously signed-in with a different account, the user signed out.
Post-conditions	The user's account data is stored in the database and the user is directed to the Start Page.
Trigger	The user does not have an account yet and would like to register to use the application.
Workflow	1. The front-end shall display the Start Page for the user. 2. The user shall press the "Sign Up" button. 3. The front-end shall display the Signup Page with fields for the user to fill out (username, email, password). 4. The user shall fill out each field and press the Sign Up button. 5. The front-end shall emit a "signup" message to the server. 6. The server shall create a User object with a unique userId, a hashed password, and associated username and email. 7. The server shall notify the frontend of successful signup.

	8. The front-end shall redirect the user to the Home Page.
Alternate Workflow	A text field is left blank: 5. The front-end shall display a message stating that a text field was left blank. The username submitted is already in use: 6. Signup shall fail and the back-end shall send an error message to the front-end. 7. The front-end shall display that the username was invalid. The email submitted is already in use: 6. Sign-up shall fail and the back-end shall send an error message to the front-end. 7. Front-end shall display that the email was invalid.

DESIGN USE-CASE 1.2: Sign-In

Priority	High - Must have feature
Status	Completed
Description	The user logs into their account using their username and password.
User Goal	The user wants to log into their account to use the application.
Desired Outcome	The user has access to the functionality of the app.
Actor	The user of the application.
Dependent Use Cases	UC 1.1
Requirements	SR 1.2
Pre-conditions	The user has an account, is not logged in, and is on the Login Page.
Post-conditions	The user is logged in.
Trigger	The user is not logged in and wants to login to access the functionality of the app.
Workflow	1. The front-end shall display the Login Page with fields for username and password. 2. The user shall fill out the fields and tap the “Login” button. 3. The front-end shall emit a “login” message to the server to start the login process. 4. The server shall authenticate the user. 5. The server shall create a session and login the user. 6. The server shall send a message to the front-end to signal successful login. 7. The front-end shall redirect the user to the Home Page.
Alternate Workflow	The username / password combination does not exist or was not registered: 5. Authentication shall fail and the server shall send

	an error message to the front-end. 6. The front-end shall display that the username was invalid.

DESIGN USE-CASE 1.3: Sign-Out

Priority	High - Must have feature
Status	Completed
Description	The user logs out of their account.
User Goal	The user wants to remove access to their account from the device, and / or wants to sign in to another account.
Desired Outcome	The user is logged out of the application.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 1.3
Pre-conditions	The user has an account and is logged in. The user has navigated to the Profile Page.
Post-conditions	The user is logged out of their account.
Trigger	The user wants to log out of their account.
Workflow	1. The front-end shall display the Home Page to the user. 2. The user shall tap the “Profile” button. 3. The front-end shall display the Profile Page. 4. The user shall tap the “Sign Out” button. 5. The front-end shall emit a “sign_out” message to the server. 6. The server shall end the user’s connection and save their data. 7. The server shall send a message to the front-end indicating log-out. 8. The front-end shall return the user to the Start Page.
Alternate Workflow	N/A

DESIGN USE-CASE 1.4: Reset Password

Priority	Lowest - Would be nice to have feature
Status	Not started
Description	The user resets their password by entering their email. An email containing the password reset link will be sent to the user.
User Goal	The user wants to reset their password, so that they can login to their account.
Desired Outcome	The user is able to login with their new password and not the old password.
Actor	The user of the application.
Dependent Use Cases	UC 1.1
Requirements	SR 1.4
Pre-conditions	The user has an account and is on the Login Page.
Post-conditions	The system shall associate the user's recovered account with their new password instead of the old password.
Trigger	The user wants to login but they have forgotten their password.
Workflow	1. The front-end shall display the Login Page. 2. The user shall tap the "forgot password" button. 3. The front-end shall display a Password Reset Page with a field for the user's email. 4. The user shall enter their email and tap the "Confirm" button. 5. The front-end shall emit a "passreset" message to the server. 6. The server shall confirm the email and send a password reset link to the user's email. 7. The user shall tap the reset link. 8. The link shall open the app and present the user with a "New Password" field. 9. The user shall enter a new password and tap the "Confirm" button.

	10. The front-end shall send a message to the server requesting password reset. 11. The new password shall be hashed and saved to the user’s User object. 12. The server shall send a success message to the front-end. 13. The front-end shall display “password reset successfully”.
Alternate Workflow	The inputted email is invalid: 6. The server shall fail to confirm the email, and send an error message to the front-end. 7. The front-end shall display an “email not found” message and prompt the user to try again. Continue from step 5 in the default workflow.

DESIGN USE-CASE 1.5: Play as Guest

Priority	Lowest - Would be nice to have feature
Status	Not Started
Description	The user accesses the functionality of the app without logging into an account.
User Goal	The user wants to play the game without logging in.
Desired Outcome	The user will be able to play the game without having an account.
Actor	The user of the application.
Dependent Use Cases	N/A
Requirements	SR 1.5
Pre-conditions	The user is on the application Start Page and is not logged into an account.
Post-conditions	The user has access to gameplay; however, none of the user's statistics or friends will be stored persistently or associated with an account.
Trigger	The user wants to play the game without creating an account.
Workflow	1. The front-end shall display the Login Page to the user. 2. The user shall tap "Play as Guest". 3. The front-end shall emit a "play_guest" message to the server. 4. The server shall create a temporary User object and send a message to the front-end. 5. The front-end shall proceed to the Home Page.
Alternate Workflow	N/A

DESIGN USE-CASE 1.6: Stay Logged In

Priority	Low - Could have feature
Status	Not Started
Description	The system remembers the user's login credentials.
User Goal	The user inputs their credentials once and the system remembers it so that the user doesn't re-enter repeatedly.
Desired Outcome	The user will be authenticated if they were previously signed in.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 1.6
Pre-conditions	The user has previously signed in to the application, and the app is closed (not backgrounded).
Post-conditions	The user is automatically signed in upon opening the app from a closed state.
Trigger	The user wants to be authenticated without entering their credentials every time they open the app from a closed state.
Workflow	1. The user shall open the app from a closed state. 2. The front-end shall display the Start Page to the user. 3. The front-end shall retrieve the login credentials from local persistent storage. 4. The front-end shall emit a "login" message to the server. 5. The server shall authenticate the user. 6. The server shall create a session and login the user. 7. The server shall send a message to the front-end to signal successful login. 8. The front-end shall redirect the user to the Home Page.

Alternate Workflow	N/A

DESIGN USE-CASE 2.1: Add a Friend

Priority	High - Must have feature
Status	Completed
Description	The user adds a friend to their friend list with the friend's username.
User Goal	The user wants to add a friend to their friend list in order to play games together in the future, and to see their match statistics.
Desired Outcome	The user's friend is added to their friends list.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 2.1
Pre-conditions	The user is logged into an account and the user's friend has an account. The user is on the Home Page.
Post-conditions	The user's friend is added to the user's friends list.
Trigger	The user wants to add a friend to their friends list.
Workflow	1. The front-end shall display the user's friends list. 2. The user shall tap the 'add friends' icon in the top right corner and enter their friend's username in the search field. 3. The front-end shall send the username to the server. 4. The server shall perform a User lookup on the username and send the User info to the front-end. 5. The front-end shall display the friend's info and an "Add" button. 6. The user shall tap the "Add" button to the right of the user they want to add as a friend.

	7. The front-end shall emit the “add_friend” message to the server. 8. The server shall add the friend to the user’s friends list and send a success message to the front-end. 9. The front-end shall refresh the user’s friends list to display the added friend.
Alternate Workflow	The username submitted is invalid: 4. The server shall perform a User lookup on the username, and send an empty response to the front-end. 5. The front-end shall display an empty search result list.

DESIGN USE-CASE 2.2: Remove a Friend

Priority	Medium - Should have feature
Status	Completed
Description	The user removes a friend from their friends list.
User Goal	The user wants to remove a friend.
Desired Outcome	The user's friend is removed from their friends list.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1
Requirements	SR 2.2
Pre-conditions	The user has an account, is logged in, and the friend is in their friends list. The user is on the Home Page.
Post-conditions	The friend is removed from the user's friends list
Trigger	The user wants to remove a friend from their friends list.
Workflow	1. The user shall tap on the "Edit Friends" button on the top of the screen. 2. The user shall tap the "Remove" button to the right of the friend they wish to remove. 3. The front-end shall emit a "remove_friend" message to the server. 4. The server shall remove the friend from the user's friend list, and remove the user from their friend's friend list. 5. The server shall send a success message to the front-end. 6. The front-end shall refresh the user's friend list.
Alternate Workflow	N/A

DESIGN USE-CASE 2.3: Block a Player

Priority	Lowest - Would be nice to have feature
Status	Completed
Description	The user blocks a friend from their friends list.
User Goal	The user wants to block a friend from their friends list.
Desired Outcome	The user successfully blocked their friend.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1
Requirements	SR 2.3
Pre-conditions	The user has an account and is logged into their account. The player the user wants to block also has an account and is on the user's friends list. The user is on the Home Page.
Post-conditions	The player the user wishes to block is listed as a blocked on the user's blocked list, and they will not be able to play together in the future.
Trigger	The user wishes to block another user.
Workflow	1. The front-end shall display the user's friends list. 2. The user shall tap the "Edit" button at the top of the screen. 3. The front-end shall display a "Remove" and "Block" button to the right of each friend in the user's friends list. 4. The user shall tap the "Block" button to the right of the friend they wish to block. 5. The front-end shall emit a "block_user" message to the server. 6. The server shall add the blocked friend's userID to the user's blocked list. 7. The server shall send a success message to the front-end. 8. The front-end shall refresh the friends list to remove the blocked friend and refresh the blocked

	list to add the new blocked friend.
Alternate Workflow	N/A

DESIGN USE-CASE 2.4: Unblock a Player

Priority	Lowest - Would be nice to have feature
Status	Completed
Description	The user unblocks a previously blocked user.
User Goal	The user wants to unblock a previously blocked user.
Desired Outcome	The previously blocked user is no longer blocked by the user and may be added as a friend again.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.3
Requirements	SR 2.4
Pre-conditions	The user has an account and is logged into their account. The user has previously blocked another user. The user is on the Home Page.
Post-conditions	The player the user wishes to unblock is no longer listed as a blocked player on the user's blocked list, and they will be able to play together in the future.
Trigger	The user wishes to unblock a player.
Workflow	1. The front-end shall display the user's friends list. 2. The user shall tap the "Blocked Players" button at the top of the screen. 3. The front-end shall emit a "get_blocked_users" message to the server. 4. The server shall retrieve the user's blocked players list and send it to front-end. 5. The front-end shall display the user's blocked players list. 6. The user shall tap the "Unblock" button to the right of the friend they wish to unblock. 7. The front-end shall send a message to the server. 8. The server shall remove the unblocked friend's userID from the user's blocked list. 9. The server shall send a success message to the front-end.

	10. The front-end shall refresh the blocked players list to remove the unblocked friend.
Alternate Workflow	N/A

DESIGN USE-CASE 2.5: Challenge a Friend to a Match

Priority	High - Must have feature
Status	Completed
Description	The user challenges a friend to a match.
User Goal	The user wants to challenge a friend to a match.
Desired Outcome	The user starts a match with a friend.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1
Requirements	SR 2.5
Pre-conditions	The friend the user wants to challenge is online and on their Friends Page. The user is on the Home Page.
Post-conditions	A match starts between the user and the friend.
Trigger	The user wishes to play a match against their friend.
Workflow	1. The user shall tap the “Challenge” button next to the friend they wish to challenge. 2. The front-end shall send a message to the server. 3. The server shall find the user associated with the selected friend’s userId and send a challenge to the friend. 4. The front-end on the friend’s instance of the app shall display the challenge from the original user. 5. The friend shall accept the challenge. 6. The front-end shall emit a “challenge_user” message to the server. 7. The server shall initiate a peer-to-peer connection between the two users using WebRTC. 8. The front-end shall proceed to displaying the game board and the match shall begin.
Alternate Workflow	The friend rejects the match offer: 6. The friend shall reject the match offer.

	7. The front-end shall send a message to the server indicating that the match was rejected. 8. The server shall abort the match attempt.

DESIGN USE-CASE 3.1: Move Avatar

Priority	High - Must have feature
Status	Completed
Description	The user moves their avatar on the board.
User Goal	The user wants to move their avatar on the board.
Desired Outcome	The user's avatar has moved.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 4.2
Requirements	SR 3.1
Pre-conditions	The user is on the Match Page, it is the user's turn, and the user is able to move.
Post-conditions	The user has made a move and their avatar on the board moved accordingly.
Trigger	It's the user's turn to move.
Workflow	1. The user shall begin their turn to move. 2. The front-end shall enable the user to swipe on the screen to move. 3. The user shall swipe in the respective direction they wish to move in (up, down, left, right). 4. The front-end shall confirm that the move the user attempted to make was valid. 5. The front-end shall update the avatar's position on the board and send a "move" message to the other player via WebRTC to indicate this change.
Alternate Workflow	The selected position was not valid: 5. The front-end shall indicate to the user that the specified position was not valid and allow the user to make another move.

DESIGN USE-CASE 3.2: Place Shape

Priority	High - Must have feature
Status	Completed
Description	The user places a shape on the board.
User Goal	The user wants to place a shape on the board.
Desired Outcome	The user placed a shape on the game board.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5
Requirements	SR 3.2
Pre-conditions	The user is on the Match Page, it is the user's turn, and the user is able to place a shape.
Post-conditions	The user has placed a shape on the board.
Trigger	It is the user's turn to place a shape.
Workflow	1. The front-end shall display the object to be placed at the top of the screen. 2. The user shall tap on the game board cell they want to place the object in. 3. The front-end shall verify that the selected position is valid. 4. The front-end shall place the object on the board and send a "place" message to the other player via WebRTC to indicate the new object.
Alternate Workflow	The object placement is invalid: 2. The front-end shall indicate to the user that an object cannot be placed in that position and allow the user to try again.

DESIGN USE-CASE 3.3: Win Match

Priority	High - Must have feature
Status	Completed
Description	The user wins the match.
User Goal	The user wants to win the match.
Desired Outcome	The user won the match.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 3.1, UC 3.2
Requirements	SR 3.3
Pre-conditions	The user is on the Match Page.
Post-conditions	The user has won the match, and the match has ended.
Trigger	The user wants to win the match and to increase their rating.
Workflow	1. The front-end shall determine that the last move made met a win-condition. 2. The front-end shall redirect both users to the PostMatch Page. 3. The front-end shall indicate to the player who met the win condition that they have won. 4. The front-end shall emit a "game_finished" message to the server. 5. The server shall log the results of the match.
Alternate Workflow	N/A

DESIGN USE-CASE 4.1: Play Locally

Priority	Low - Could have feature
Status	Completed
Description	The user plays a match locally.
User Goal	The user wants to play a local match.
Desired Outcome	The user started a local match.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 4.1
Pre-conditions	The user has an account and is logged into that account. The user is on the Home Page.
Post-conditions	The user is in a local match, playing as both P1 and P2.
Trigger	The user wants to play a match locally.
Workflow	1. The user shall tap on the "Local Play" button. 2. The front-end shall initiate a match with player 1 and player 2 defined as the user. 3. The front-end shall proceed to displaying the Match Page and the single-player match shall begin.
Alternate Workflow	N/A

DESIGN USE-CASE 4.2: Play Random Opponent

Priority	Low - Could have feature
Status	Complete
Description	The user pairs up with a random user based on rating and a match is started between them.
User Goal	The user wants to be in a match against a random user of similar skill level.
Desired Outcome	The user paired up with a random user for a match.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 4.2
Pre-conditions	The user is on the Friends Page and another user has pressed the “Random Play” button on the Friends Page.
Post-conditions	The user is in a match with a random user.
Trigger	The user wants to play a match against a random user.
Workflow	1. The user shall tap the “Random Play” button. 2. The front-end shall emit a “match_random” message to the server. 3. The server shall add the player to the matchmaking queue. 4. When an opponent is found, the server shall emit a “match_initiate” message and initiate a match between the players via WebRTC.
Alternate Workflow	N/A

DESIGN USE-CASE 4.3: Analyze Aggregate Match Statistics

Priority	Lowest - Would be nice to have feature
Status	Not Started
Description	The user views a collection of data and statistics gathered from the user's previous matches.
User Goal	The user wants to view their aggregate match statistics.
Desired Outcome	The user accessed their match statistics.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 4.3
Pre-conditions	The user has an account, is logged in, and is on the Home Page.
Post-conditions	The user has been redirected to the Statistics Page.
Trigger	The user is interested in how they are doing in the game.
Workflow	1. The user shall tap on the "Profile" button. 2. The front-end shall display the Profile Page. 3. The user shall tap on the "Statistics" button. 4. The front-end shall emit a "get_statistics" message to the server. 5. The server shall query the database to fetch aggregate match statistics. 6. The server shall send the statistics to the front-end. 7. The front-end shall display the Statistics Page and populate it with the user's statistics.
Alternate Workflow	N/A

DESIGN USE-CASE 4.4: Rematch After a Match Ends

Priority	Low - Could have feature
Status	Complete
Description	The user requests a rematch after a match ends.
User Goal	The user wants to rematch against their opponent.
Desired Outcome	The user rematches against their opponent.
Actor	The user who has just finished a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 3.1, UC 3.2, UC 3.3, UC 4.2
Requirements	SR 4.4
Pre-conditions	The user and their opponent are on the PostMatch Page.
Post-conditions	The user and their opponent are on the Match Page in a new match.
Trigger	The user's match has just ended and the user wants a rematch for another opportunity to earn rating from their opponent.
Workflow	1. The front-end shall display the PostMatch Page to both players. 2. The user shall tap the "Rematch" button at the bottom of the screen. 3. The front-end shall send a "rematch" message to the opponent via WebRTC that the user has requested a rematch. 4. The front-end shall display a rematch request message to the opponent. 5. The opponent shall tap the "Rematch" button. 6. The front-end shall send a message to the server. 7. The server shall log the results of the previous match. 8. The server shall continue the peer-to-peer connection between the two users using WebRTC.

	9. The front-end shall proceed to displaying the game board and a new match shall begin.
Alternate Workflow	The opponent ignores the rematch request: 5. The opponent shall tap the “Quit” button. 6. The front-end shall terminate the connection between the two players. 7. The front-end shall send a “game_finished” message to the server. 8. The server shall log the results of the match.

DESIGN USE-CASE 4.5: Prematurely End a Match

Priority	Medium - Should have feature
Status	Complete
Description	The user leaves in the middle of an ongoing match.
User Goal	The user wants to leave an ongoing match.
Desired Outcome	The user left the match.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5
Requirements	SR 4.5
Pre-conditions	The user is in a match with another user, and on the Match Page.
Post-conditions	The user is no longer in a match, and on the Friends Page.
Trigger	The player wishes to leave in the middle of a match.
Workflow	1. The user shall tap the “Quit” button. 2. The front-end shall send a “close” message to the other user via WebRTC. 3. The front-end shall send a message to the server. 4. The server shall record a loss for the user. 5. The front-end shall return the user and the opponent to the Home Page.
Alternate Workflow	N/A

DESIGN USE-CASE 4.6: Chat With Opponent

Priority	Lowest - Would be nice to have feature
Status	Not Started
Description	The user communicates with the opponent via text during a match.
User Goal	The user wants to communicate with the opponent via text during a match.
Desired Outcome	The user chats with the opponent.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 4.2
Requirements	SR 4.6
Pre-conditions	The user is in a match with another user, and on the Match Page.
Post-conditions	The chat is updated with new messages from either player.
Trigger	The user wants to say something to say to their opponent.
Workflow	1. The user shall tap the chat bar on the bottom of the screen. 2. The front-end shall open and expand the chat interface. 3. The user shall input the desired message and tap send. 4. The front-end shall send a “chat” message via WebRTC. 5. The front-end on the other player’s instance shall update the chat to display the new message.
Alternate Workflow	Cancel Message 3. The user shall tap the “x” button on the top-right corner of the chat screen to close the extended conversation screen.

DESIGN USE-CASE 4.7: Watch a Replay of a Match

Priority	Lowest - Would be nice to have feature
Status	Not Started
Description	The user watches a replay of a match they played.
User Goal	The user wants to watch a replay of a match they played.
Desired Outcome	The user viewed a replay of a previous match.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 4.2
Requirements	SR 4.7
Pre-conditions	The user has played at least one match. The user is on the Profile Page.
Post-conditions	The user is redirected to the MatchReplay Page, where they view a replay of a match.
Trigger	The user wants to learn from previous mistakes and improve their strategy.
Workflow	1. The user shall tap the "Match History" button. 2. The front-end shall send a message to the server. 3. The server shall query the database to get old match replays. 4. The server shall send the replays to the front-end. 5. The front-end shall display the available replays on the MatchHistory Page. 6. The user shall select a replay. 7. The front-end shall navigate to the MatchReplay Page and load the replay.
Alternate Workflow	N/A

DESIGN USE-CASE 4.8: Review Moves Made During Match

Priority	W - Would be nice to have feature
Status	Not Started
Description	The user views the moves made during the current match.
User Goal	The user wants to review the moves made during a match.
Desired Outcome	The user viewed previous moves made.
Actor	The user in a match.
Dependent Use Cases	UC 1.1, UC 1.2, UC 2.1, UC 2.5, UC 4.2
Requirements	SR 4.8
Pre-conditions	The user is on the Match Page.
Post-conditions	The user is redirected to the MatchMoves Page.
Trigger	The user or the opponent has made a move.
Workflow	1. The user shall tap on the “Previous Moves” button. 2. The front-end shall redirect the user to the MatchMoves Page and display an ordered list of previous moves made in the match.
Alternate Workflow	N/A

DESIGN USE-CASE 5.1: Change Player Avatar

Priority	L - Could have feature
Status	Completed
Description	The user changes their in-game avatar.
User Goal	The user wants to change their in-game avatar.
Desired Outcome	The user changed their in-game avatar.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 5.1
Pre-conditions	The user is on the Profile Page.
Post-conditions	The user's avatar has been changed to an avatar of their choice.
Trigger	The user wants to change their avatar.
Workflow	1. The user shall tap on either the "<" or ">" buttons next to their current avatar. 2. The front end shall display the selected avatar and send a "set_avatar" message to the server with the new avatar. 3. The server shall send a success message and update the user's respective avatar.
Alternate Workflow	N/A

DESIGN USE-CASE 5.2: Change App Icon

Priority	W - Would be nice to have feature
Status	Completed
Description	The user changes the app's icon on their device.
User Goal	The user wants to change the app's icon on their device.
Desired Outcome	The user changed the app's icon on their device.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 5.2
Pre-conditions	The user is on the Profile Page.
Post-conditions	The user's app's icon has been changed to their avatar.
Trigger	The user wants to change their app icon to show off their avatar of choice.
Workflow	1. The user shall tap on the "Set App Icon" button at the bottom of the screen. 2. The front end shall set the user's avatar image as the app's icon image on the user's device.
Alternate Workflow	N/A

DESIGN USE-CASE 5.3: Play Tutorial

Priority	W - Would be nice to have feature
Status	Completed
Description	The user learns the rules of the game.
User Goal	The user wants to learn the rules of the game.
Desired Outcome	The user is familiar with the game rules.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 5.3
Pre-conditions	The user is on the Home Page.
Post-conditions	The user is on the Home Page.
Trigger	The user is unfamiliar with the rules of the game.
Workflow	1. The user shall tap the “Tutorial” button. 2. The front-end shall display the predefined tutorial stages. 3. The user shall complete the stages in the tutorial. 4. The front-end shall display a message at the end of the tutorial, notifying the user that they have completed the tutorial. 5. The user shall tap the “Finish” button. 6. The front-end shall display the Home Page.
Alternate Workflow	User quits the tutorial: 3. The user shall tap the “Quit” button at the bottom of the Tutorial Page. 4. The front-end shall display a confirmation popup message. 5. The user confirms their decision. 6. The front-end shall display the Home Page.

DESIGN USE-CASE 5.4: View Leaderboard

Priority	Low - Could have feature
Status	Complete
Description	The user views the leaderboard to see where they rate in the game in comparison to other players.
User Goal	The user wants to check their own or another player's rating.
Desired Outcome	The user accessed the leaderboard and viewed player ratings.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 5.4
Pre-conditions	The user is on the Home Page.
Post-conditions	The user has been redirected to the Leaderboard Page.
Trigger	The user wants to see where they rate in the game in comparison to other players.
Workflow	1. The user shall tap the "Leaderboard" button. 2. The front-end shall send a "top_players" message to the server 3. The server shall query the database to fetch the leaderboard. 4. The server shall send the leaderboard data to the front-end. 5. The front-end shall load the Leaderboard Page and populate it with leaderboard data.
Alternate Workflow	N/A

DESIGN USE-CASE 5.5: View Rating

Priority	Low - Could have feature
Status	Complete
Description	The user checks their rating.
User Goal	The user wants to check their rating.
Desired Outcome	The user checked their rating.
Actor	The user of the application.
Dependent Use Cases	UC 1.1, UC 1.2
Requirements	SR 5.5
Pre-conditions	The user is on the Home Page.
Post-conditions	The user is on the Profile Page.
Trigger	The user wants to know what their personal rating is.
Workflow	1. The player shall tap the "Profile" button. 2. The front-end shall retrieve the user's information by emitting a "whoami" message to the server. 3. The front-end shall load and display the Profile Page with the user's rating.
Alternate Workflow	N/A