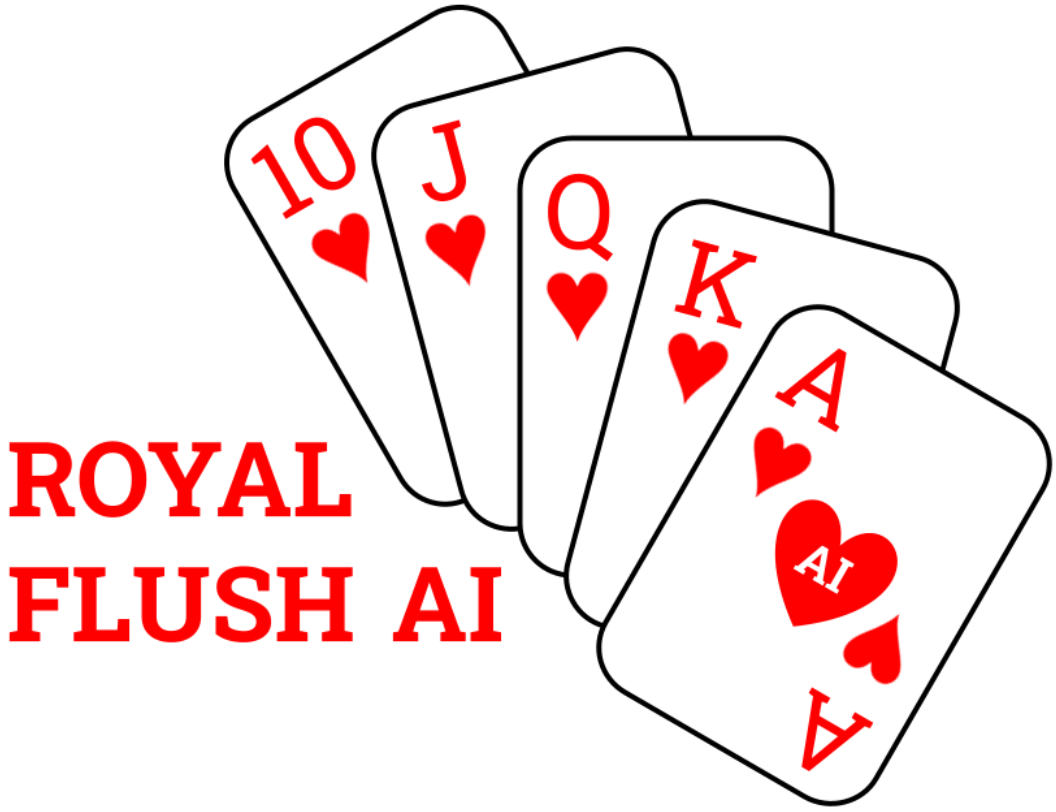
RoyalFlushAI

Design Document



TEAM 28

Andrei Deaconescu

Andrew Ge

Jacob Mobley

Raymond Wang

Table of Contents

Purpose

Currently, there is no reliable way to play poker against an AI. This project aims to solve this by training an AI model to play poker, enabling a way to play poker against other people and delivering a variety of other games such as blackjack and roulette.

With this, we will establish a game platform that is easy to use, entertaining, and reliable. Additionally, our platform will use a currency system that is entirely within the app and does not require the use of actual money to bring a safe gaming environment to everyone.

Functional Requirements

Account Management

1. As a user, I would like to create an account with a username and password.
2. As a user, I would like to be able to change my username.
3. As a user, I would like to be able to change my password.
4. As a user, I would like to have my progress saved when I log out.
5. As a user, I would like to add a bio to my profile.
6. As a user, I want to be able to customize my profile with an avatar.

Gameplay

1. As a user, I would like to play against the Texas Hold'Em AI.
2. As a user, I would like to play against other players.
3. As a user, I would like to play against the AI and other players at the same time.
4. As a user, I would like to be able to play minigames to earn points.
5. As a user, I would like to be able to start a game of Blackjack.
6. As a user, I would like to be able to start a game of Roulette.
7. As a user, I would like to choose between singleplayer and multiplayer.
8. As a user, I would like to be able to choose the difficulty of AI.
9. As a user, I would like to buy back in if I lose all the chips from my current buy-in.
10. As a user, I would like for the rules of certain games to be customized, such as the number of decks used in blackjack.
11. As a user, I want to be able to participate in tournaments with other players.
12. As a user, I want to be able to participate in single-player tournaments against AI opponents.
13. As a user, I want the buy-in point values to translate to values for different colored chips.
14. As a user, I would like there to be a settings menu.

15. As a user, I would like there to be a timer for each turn.
16. As a user, I would like to start with some currency upon account creation.
17. As a user, I want to be alerted when it is my turn to play.
18. As a user, I want to be able to easily play the game and perform actions such as folding, raising, calling, etc.
19. As a user, I want to easily see who big and small blind is.
20. As a user, I want the ability to split the pot.
21. As a user, I want to easily understand what the total pot currently at play is.
22.
23. As a user, I want to see exactly how many chips and what type each player has.
24. As a user, I want to clearly see the winner of each round.

Design

1. As a user, I would like to play using some nicely designed cards.
2. As a user, I would like the UI to emulate a casino setting (have a green, poker table-like background)
3. As a user, I would like there to be background music.
4. As a user, I would like there to be sound effects.
5. As a user, I want the actual chips to look appealing.

Social Interaction

1. As a user, I would like to create private games.
2. As a user, I want to see a leaderboard of the highest-scoring users.
3. As a user, I would like to chat with other players.
4. As a user, I would like to be able to emote.
5. As a user, I would like to be able to add friends.

Compatibility

1. As a user, I would like to play on the website from my computer
2. As a user, I would like to be able to play on the website from my phone.
3. As a user, I would like to be able to play on an Android device if time allows.
4. As a user, I would like to be able to play on an iOS device if time allows.
5. As a user, I would like to be able to download the app on my computer if time allows.
6. As a user, I would like to play with people who use different platforms.

Game Assistance

1. As a user, I want to be able to earn achievements for completing specific tasks or milestones.

2. As a user, I want to be able to review my hand history to analyze my gameplay.
3. As a user, I want a guide on what hands are available and their rankings.
4. As a user, I want access to a help menu to clarify certain parts of the game.
5. As a user, I would like to see my past scores.
6. As a user, I would like a tutorial to help me learn the game, if time allows

Non-Functional Requirements

Security

1. As a developer, I would like to set up a Google Firebase authentication system to store user credentials.
2. As a developer, I would like to store a hashed version of the password for maximum security.
3. As a developer, I would like to make sure that players can only access each other's username, avatar, and bio.
4. As a developer, I would like to introduce a CAPTCHA system to ensure there are no robots trying to log onto our platform.

Usability

1. As a user, I want the game to be easily navigated.
2. As a user, I want to have easily accessible buttons to perform any game move.
3. As a user, I would like to have a support menu where I can reach out to the developers in case of any bugs/issues.

Response Time

1. As a user, I would like to be able to navigate the platform with little to no lag.
2. As a user, I would like to access any game within 1 sec.
3. As a user, I would like to perform in-game moves within 200ms.
4. As a user, I would like to find and join a multiplayer game within 15 sec.

Hosting

1. As a developer, I would like the platform to be easily accessible, users being able to view it in the browser.
2. As a developer, I would like to develop a React server which can run actual games.

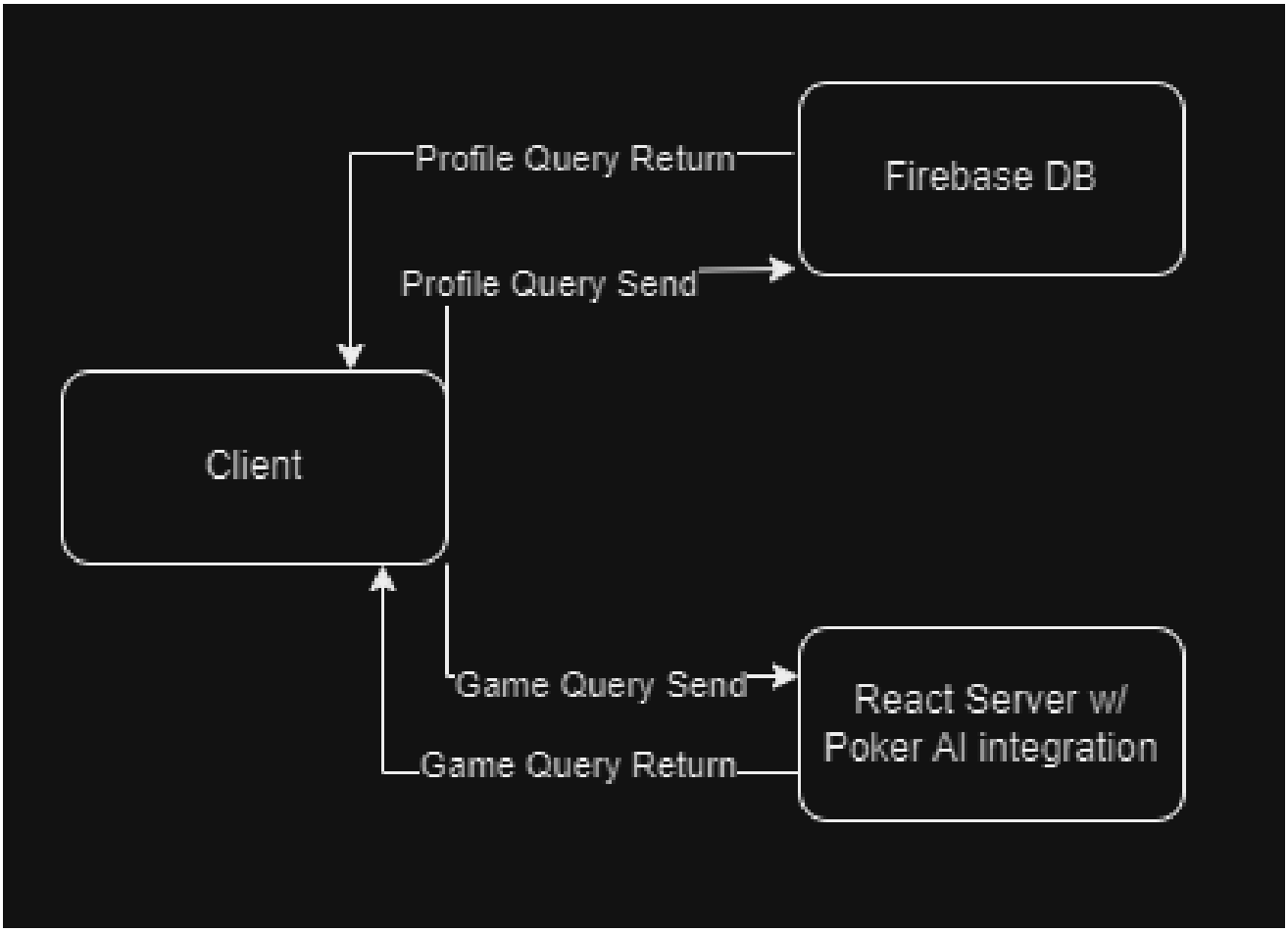
3. As a developer, I would like to use Google's Firebase database to store details individual to the user, such as currency and top scores, in addition to the login credentials.

Design Outline

Our platform has two main components: the Firebase database and the React server. When a client logs onto our server, they have the option to either create an account or log in—both options will be handled by the Firebase database. Upon login, Firebase will load all the data pertaining to the user (avatar, bio, currency, scores, achievements, etc.).

The other part of our platform is the actual playing of games. The main Poker game can either be singleplayer (against AI) or multiplayer (either exclusively against other players or a combination of players and AI). The other minigames will be exclusively singleplayer, only playable against AI. All the actual games will be handled within the React server, due to its minimal latency. We will implement Python integration in the server as well for the Poker AI to work after it has been trained.

High-Level Diagram



1. Client
    - The client can send queries to either the Firebase database upon logging into the platform, or the React server
    - There are only two endpoints the client will be able to access on the Firebase DB (fetch and store)
    - On the React server, on the other hand, there are many endpoints the client can access, ranging from selecting a game, performing moves within game, etc.

2. Firebase Database
    - The Firebase database holds all the user details, ranging from login credentials to currency, scores, friends, etc.

- The purpose of the database is to validate user authentication and load user details
- Whenever user details change (user gains/loses currency, user adds/remove friends, etc.) the React server sends a reply to the client, and the client automatically updates the details in the Firebase database appropriately

3. React Server
    - This is the main unit that will handle everything game-related, ranging from multiplayer hosting to AI usage within the Poker game
    - The client can send a variety of queries ranging from selecting a game to performing in-game moves
    - This shall also be user for general web hosting; any modifications that occur pertaining to user settings shall go through the React server first, then they will be sent to the client and the client will send them to the Firebase database

Class Diagram

This class diagram represents the structure and relationships in a multi-game application, which includes various game modes such as Poker, Blackjack, Plinko, and Roulette, as well as supporting components like user management, betting, and gameplay interactions.

The abstract class BaseGameManager provides common functionality for all game modes, such as managing the list of players and tracking the game state (e.g., waiting for players, in progress, or game ended). Each game mode, including PokerGameManager, BlackjackGameManager, PlinkoGameManager, and RouletteGameManager, extends this base class and handles the specific mechanics for that game.

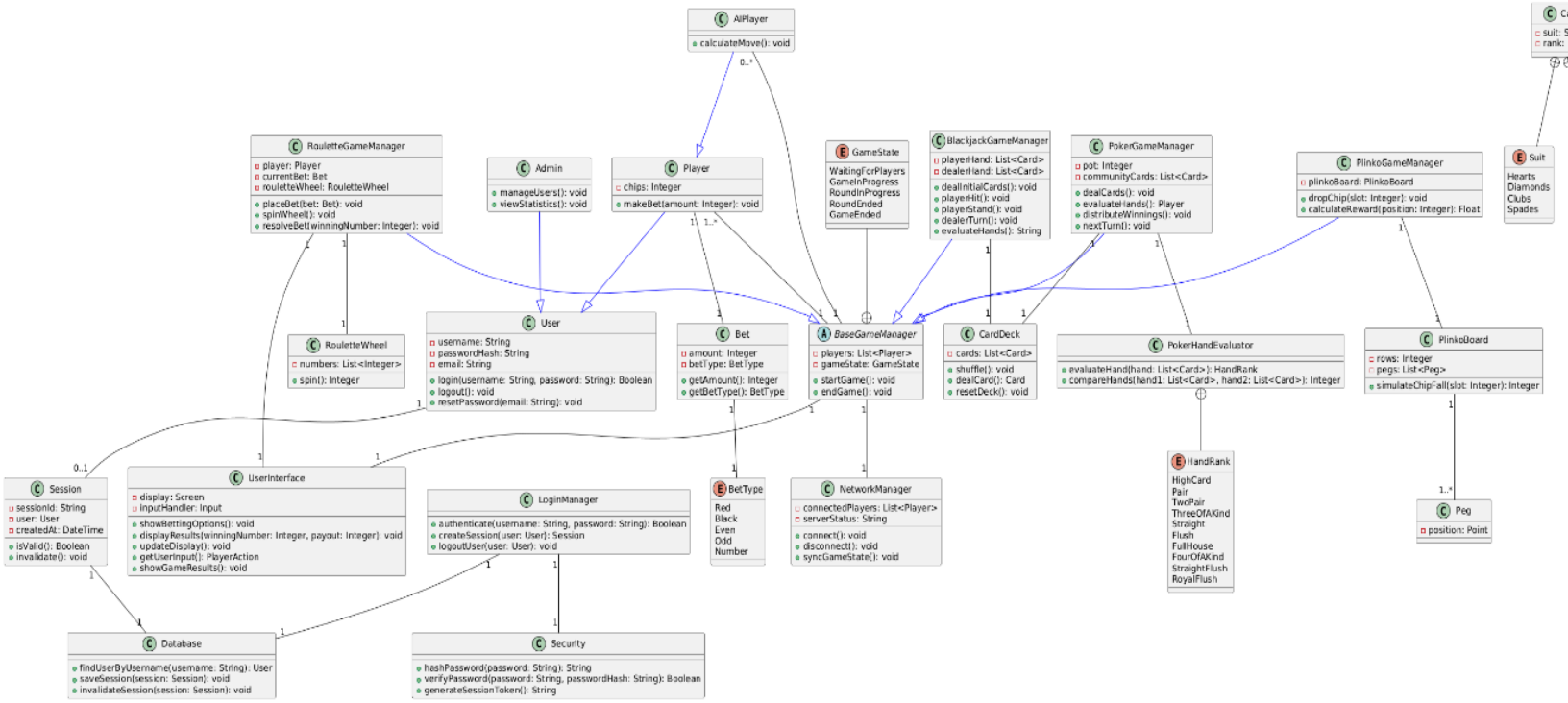
The diagram includes shared components that support user sessions and authentication. The Session class manages user sessions, including the session ID, user information, and validity status. The LoginManager handles user authentication, creating sessions, and logging users out, while the Security class is responsible for hashing and verifying passwords and generating secure session tokens. All user and session data is stored and retrieved using the Database class, which provides methods for finding users, saving sessions, and invalidating sessions.

The NetworkManager manages multiplayer connections and game state synchronization across players, ensuring smooth communication and gameplay in multiplayer modes. The UserInterface class is responsible for displaying the game state and results to the player, providing the necessary options for betting, showing game outcomes, and interacting with the game logic.

The Player class tracks attributes like chips and includes methods for placing bets. The AIPlayer inherits from Player and includes logic for automated decision-making, allowing for computer-controlled opponents.

The User class is a basic encapsulation of standard login information. Admin class extends the User class and adds administrative capabilities, such as managing users and viewing statistics, while the User class handles authentication, including logging in, logging out, and resetting passwords.

The RouletteWheel class simulates the spinning of the roulette wheel and provides a method to generate a winning number, using the Bet class to manage bets. The PlinkoBoard class manages the layout of pegs in the Plinko game, simulating chip drops and determining where the chip lands. Each Plinko board contains multiple Peg objects that represent the physical pegs in the game. The PokerGameManager manages poker-specific elements like the pot and community cards, while BlackjackGameManager handles player and dealer hands, processing actions like hitting or standing.


AllPlayer
calculateMove(): void
RouletteGameManager
player: Player
currentBet: Bet
rouletteWheel: RouletteWheel
placeBet(bet: Bet): void
spinWheel(): void
resolveBet(winningNumber: Integer): void
Admin
manageUsers(): void
viewStatistics(): void
Player
chips: Integer
makeBet(amount: Integer): void
GameState
WaitingForPlayers
GameInProgress
RoundInProgress
RoundEnded
GameEnded
BlackjackGameManager
playerHand: List<Card>
dealerHand: List<Card>
dealInitialCards(): void
playerHit(): void
playerStand(): void
dealerTurn(): void
evaluateHands(): String
PokerGameManager
pot: Integer
communityCards: List<Card>
dealCards(): void
evaluateHands(): Player
distributeWinnings(): void
nextTurn(): void
PlinkoGameManager
plinkoBoard: PlinkoBoard
dropChip(slot: Integer): void
calculateReward(position: Integer): Float
Suit
Hearts
Diamonds
Clubs
Spades
RouletteWheel
numbers: List<Integer>
spin(): Integer
User
username: String
passwordHash: String
email: String
login(username: String, password: String): Boolean
logout(): void
resetPassword(email: String): void
Bet
amount: Integer
betType: BetType
getAmount(): Integer
getBetType(): BetType
BaseGameManager
players: List<Player>
gameState: GameState
startGame(): void
endGame(): void
CardDeck
cards: List<Card>
shuffle(): void
dealCard(): Card
resetDeck(): void
PokerHandEvaluator
evaluateHand(hand: List<Card>): HandRank
compareHands(hand1: List<Card>, hand2: List<Card>): Integer
PlinkoBoard
rows: Integer
pegs: List<Peg>
simulateChipFall(slot: Integer): Integer
Session
sessionId: String
user: User
createdAt: DateTime
isValid(): Boolean
invalidate(): void
UserInterface
display: Screen
inputHandler: Input
showBettingOptions(): void
displayResults(winningNumber: Integer, payout: Integer): void
updateDisplay(): void
getUserInput(): PlayerAction
showGameResults(): void
LoginManager
authenticate(username: String, password: String): Boolean
createSession(user: User): Session
logoutUser(user: User): void
BetType
Red
Black
Even
Odd
Number
NetworkManager
connectedPlayers: List<Player>
serverStatus: String
connect(): void
disconnect(): void
syncGameState(): void
HandRank
HighCard
Pair
TwoPair
ThreeOfAKind
Straight
Flush
FullHouse
FourOfAKind
StraightFlush
RoyalFlush
Peg
position: Point
Database
findUserByUsername(username: String): User
saveSession(session: Session): void
invalidateSession(session: Session): void
Security
hashPassword(password: String): String
verifyPassword(password: String, passwordHash: String): Boolean
generateSessionToken(): String

Sequence Diagrams

The sequence of events will elaborate on the interactions between the client, the Firebase database, and the React server.

First, the players will open the app, and they will have the option to either log in or create an account.

On creating an account, the username and password will be stored into the Firebase database, and the players will be faced with a screen where they will have to set their bio and avatar. These details will also be stored in the Firebase database.

When logging in, the user enters the credentials, and they will be sent to Firebase. Firebase will validate the username and password, and it will load all the other user data into the client.

Subsequently, when the client is logged in, they can select a game to play. For minigames such as blackjack or roulette, they will be able to select the game, but these minigames do not offer a multiplayer option. The client will essentially play against an algorithm hosted on the React server.

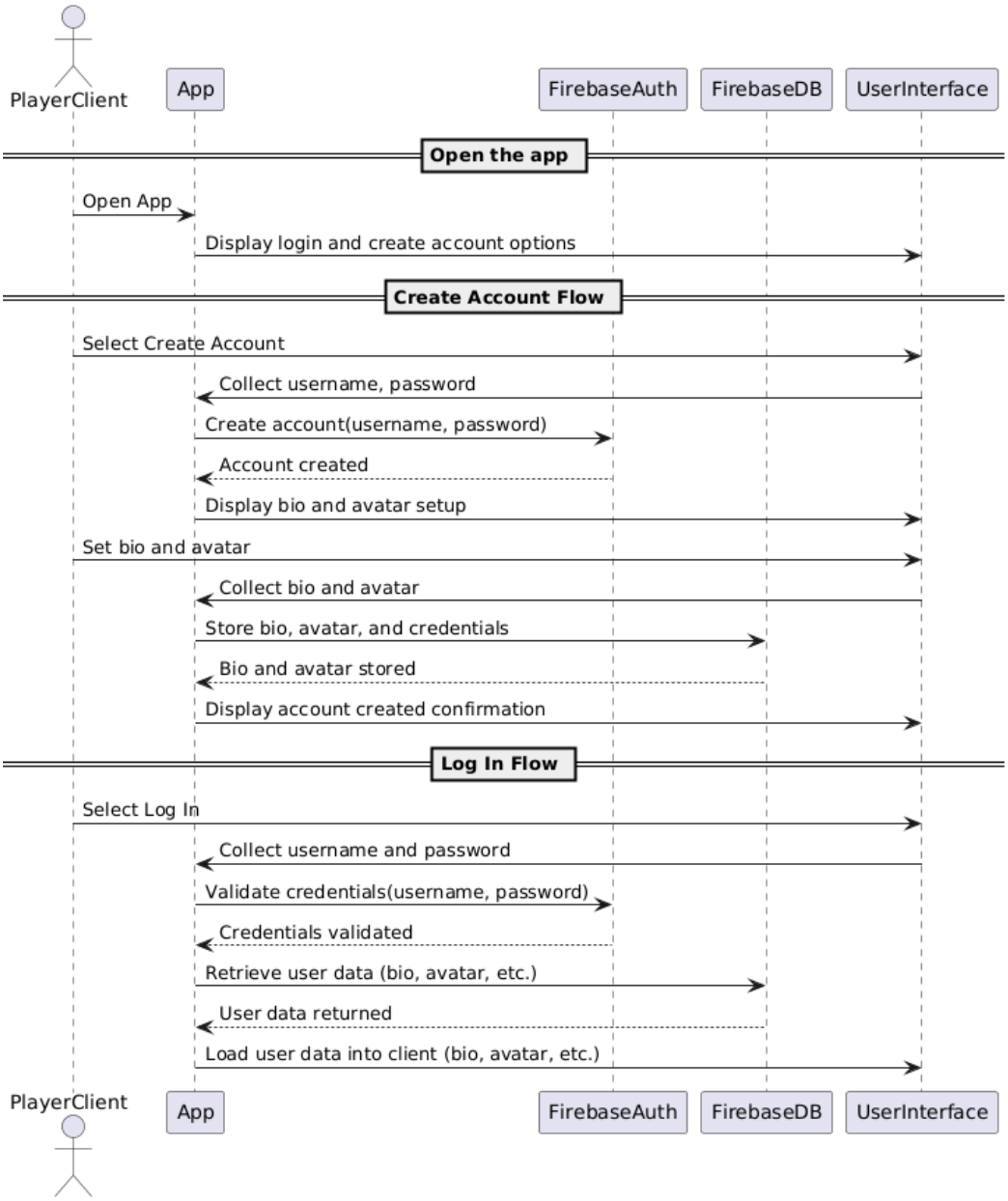
For the game of poker, however, the user will have the option to either play against other people in a multiplayer mode or play singleplayer against AI. If the multiplayer does not reach the required number of people, the remaining slots will be filled with AI bots. All of this, ranging from the multiplayer hosting to AI usage, will be stored within the React server.

Within the poker game, the client will have a variety of options including raise, check, and fold; all these calls will also be handled within the React server.

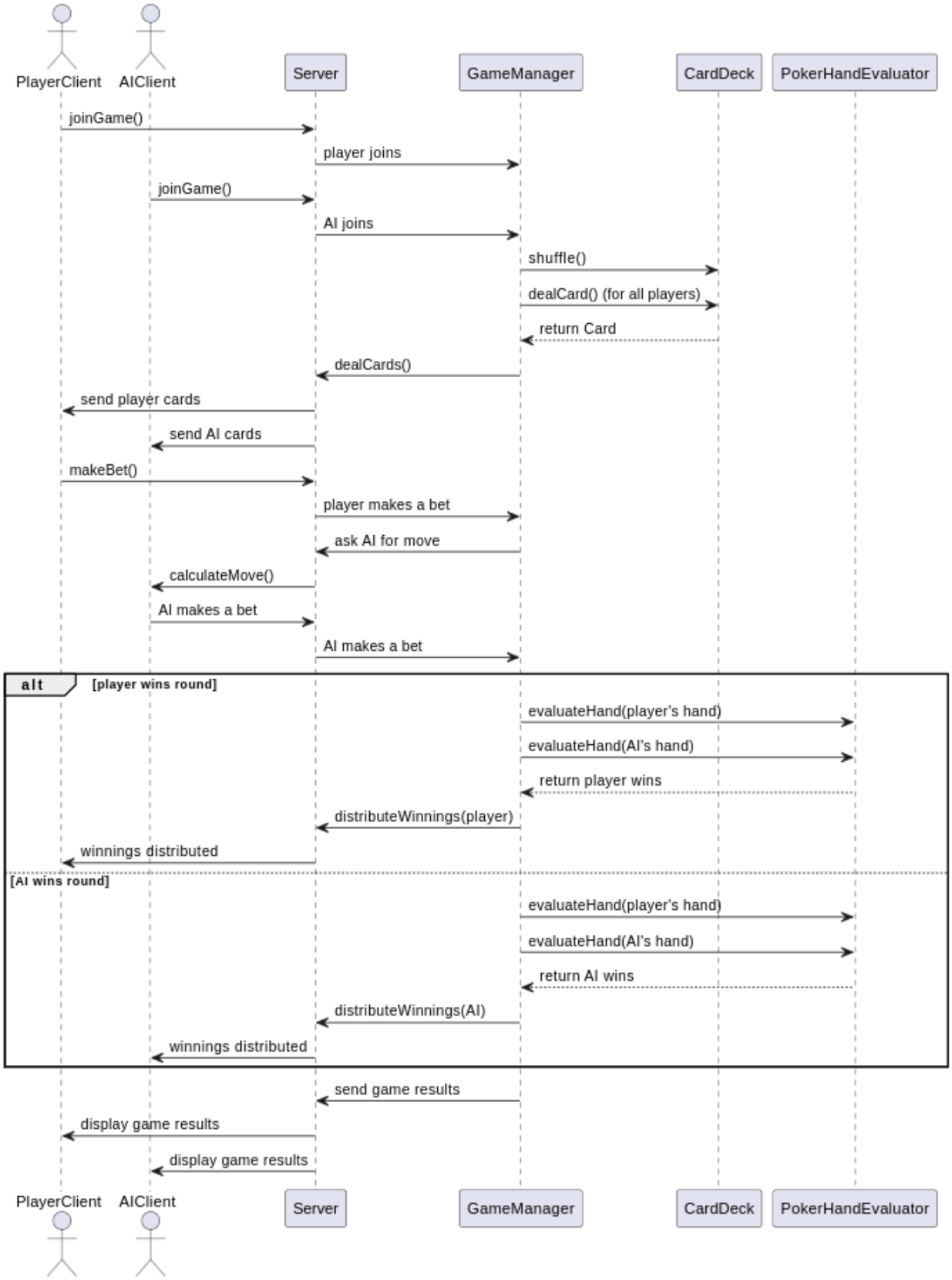
When a poker game ends and a winner is selected, the React server will end the game and the Firebase database will update with the new user details (currency gained / lost, new high scores, etc.).

A detailed diagram of the sequence of events can be found below:

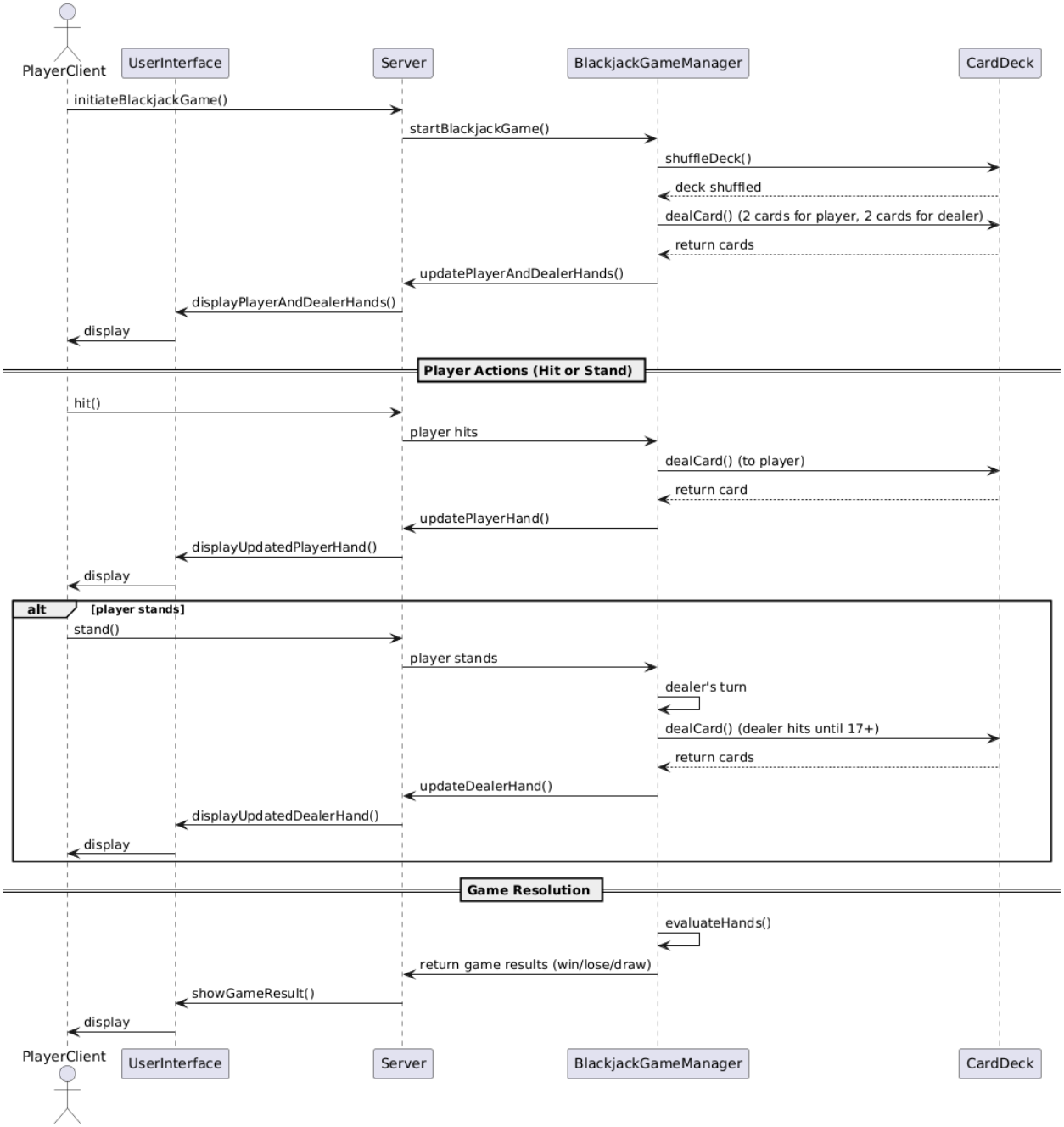
Login Sequence



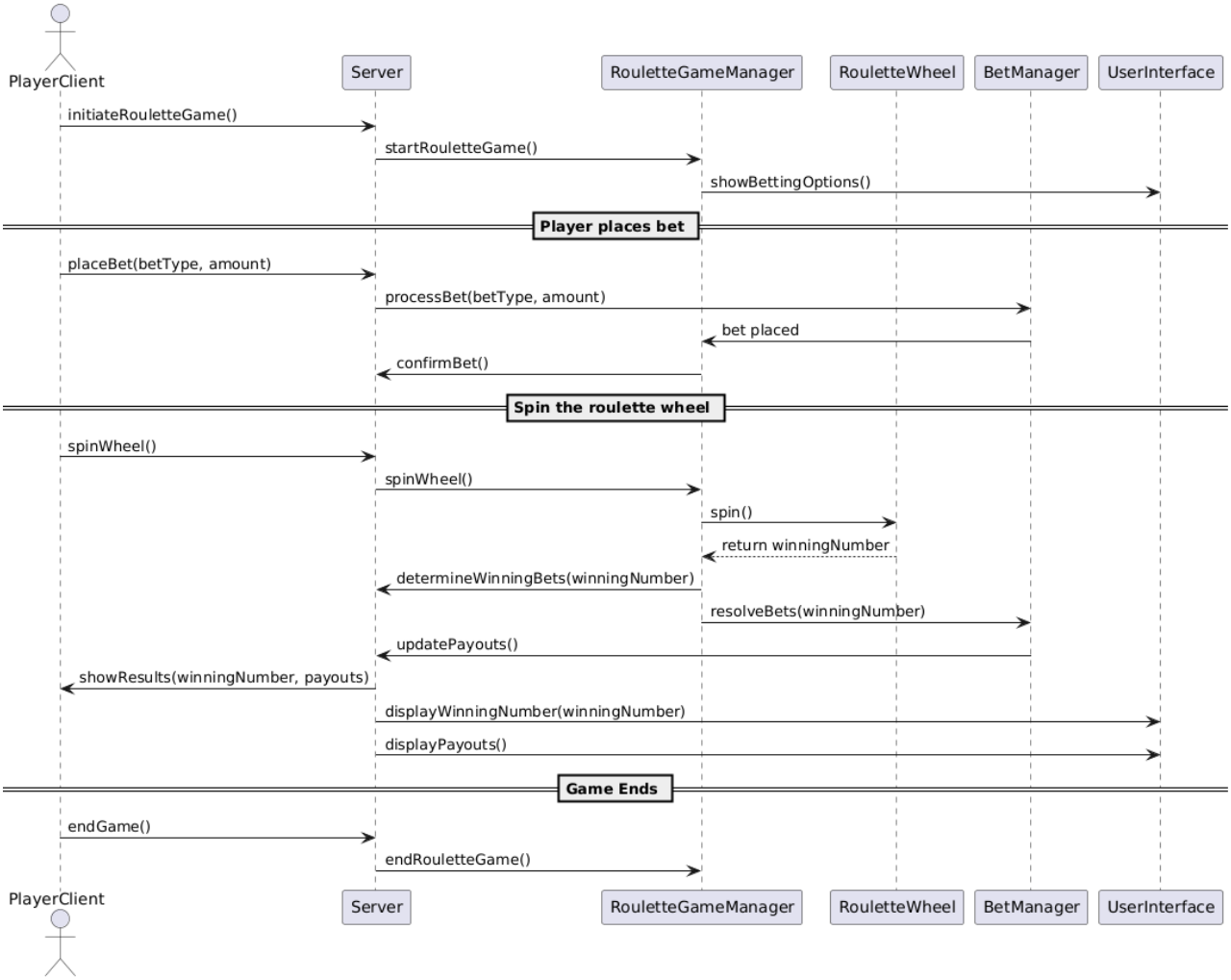
Poker Diagram



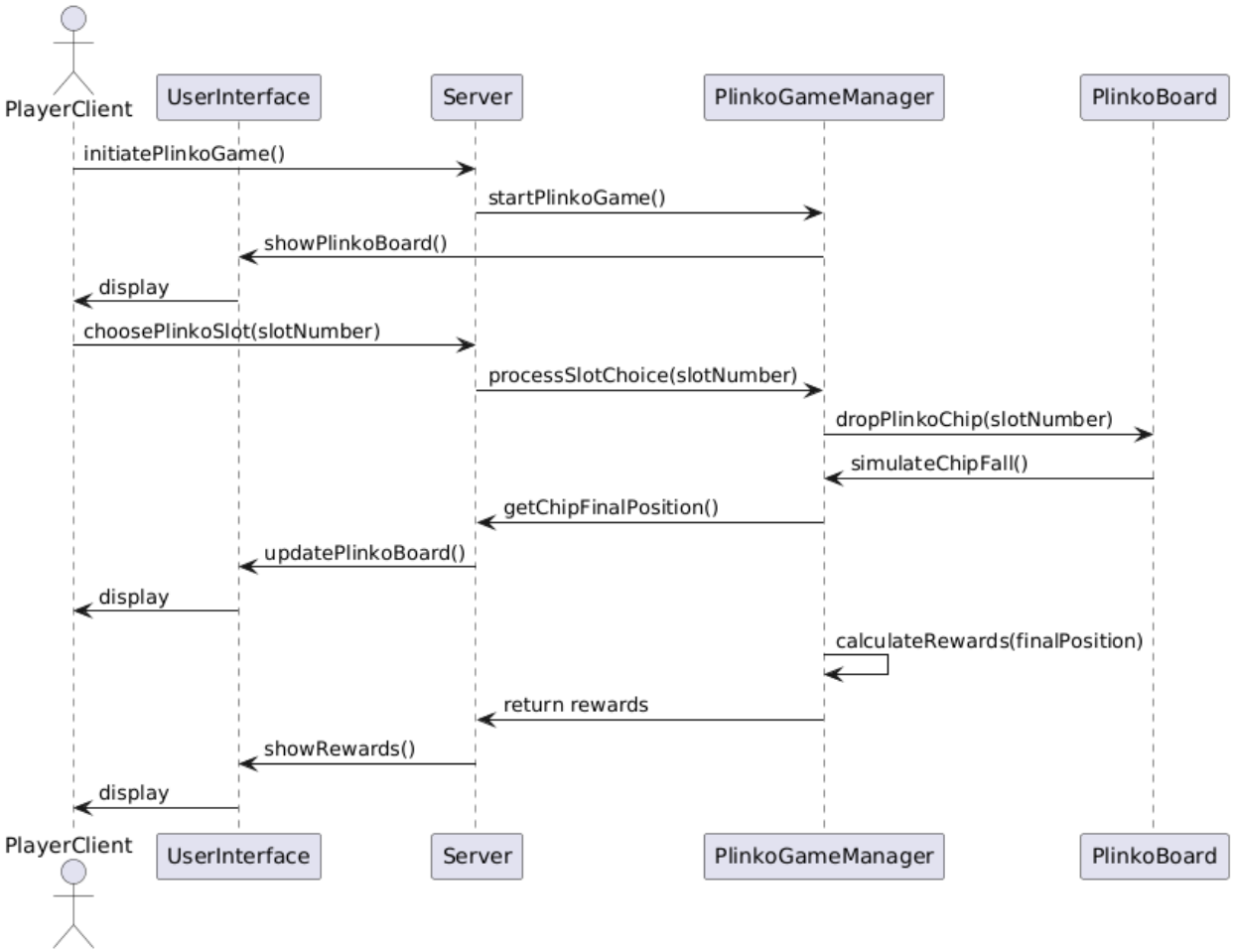
Blackjack Diagram



Roulette Diagram



Plinko Diagram



Action Diagrams

The first action diagram represents the experience of the user as they attempt to login. When the user opens the application, they are greeted with a login screen, prompting them for username/password. The application then sends the credentials to firebase for authentication. The process then diverges based on whether the credentials are valid. If the credentials are valid, the app retrieves the player's profile data (such as bio and avatar) from the Firebase Database. This data is loaded into the client, and the player is presented with their profile and the home screen. If the credentials are invalid, the app displays a login error, prompting the player to re-enter their credentials. The flow repeats until the player successfully logs in or exits the app.

The second action diagram outlines the process players go through when selecting and playing different games in a multi-mode gaming app. When the player opens the app, they are presented with several game options: Poker, Blackjack, Plinko, and Roulette. Once the player picks a game, the diagram branches off into different paths based on which game they chose.

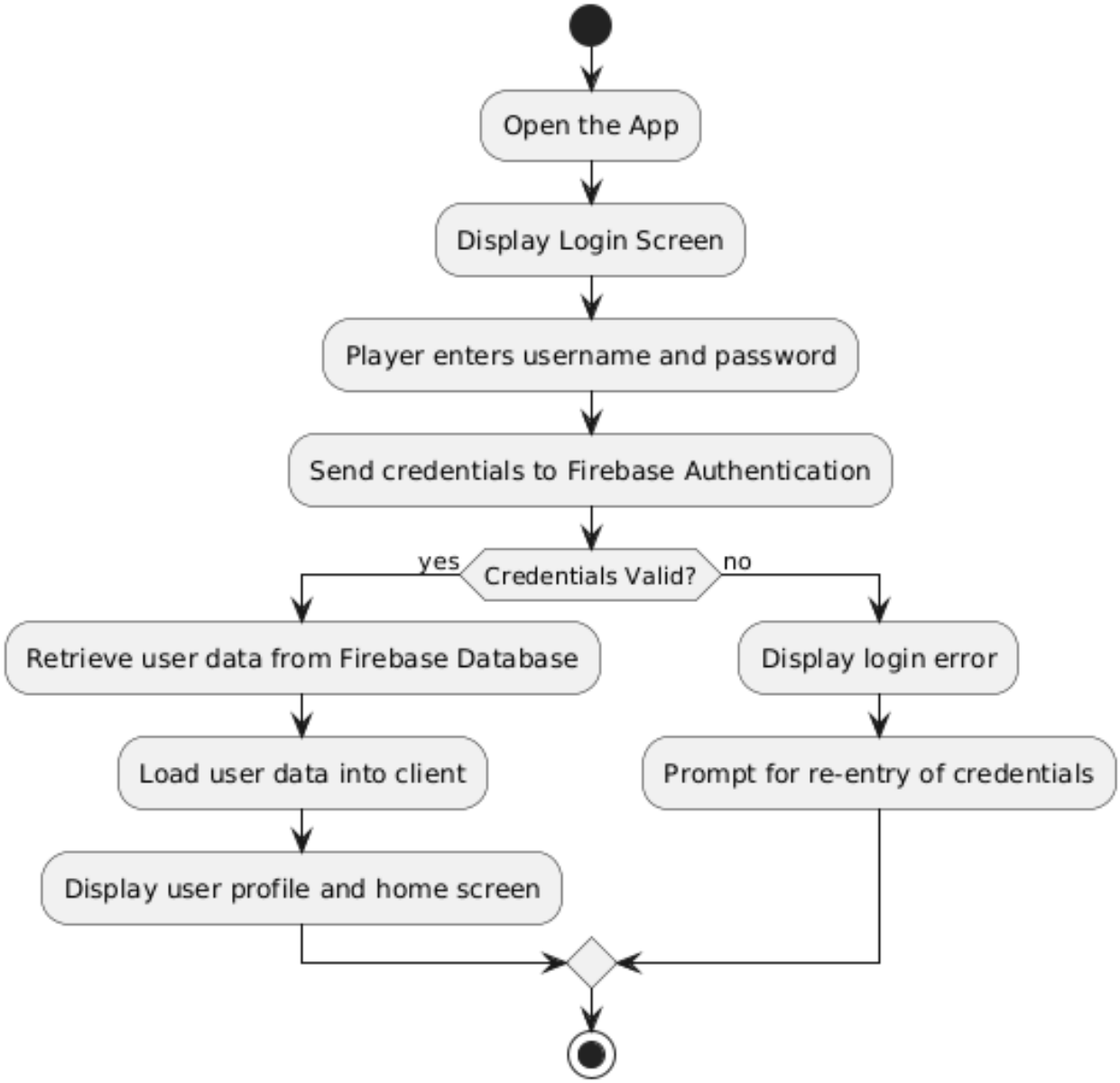
For Poker, the process starts by shuffling the deck and dealing cards to the players. Each player takes their turn and decides to either bet, check, or fold. If a player folds, they are removed from the round. If they continue playing, the next player takes their turn. After all moves are made, the game evaluates the hands, distributes the winnings, and prepares for the next round. The player is then asked if they want to keep playing or end the game.

In Blackjack, the game begins by shuffling the deck and dealing cards to both the player and the dealer. The player can then decide to hit (draw another card) or stand. If the player hits, they receive a card until they decide to stand. Once the player stands, the dealer draws cards until they reach at least 17. The hands are then evaluated, results are displayed, and the player is asked whether they want to continue or end the game.

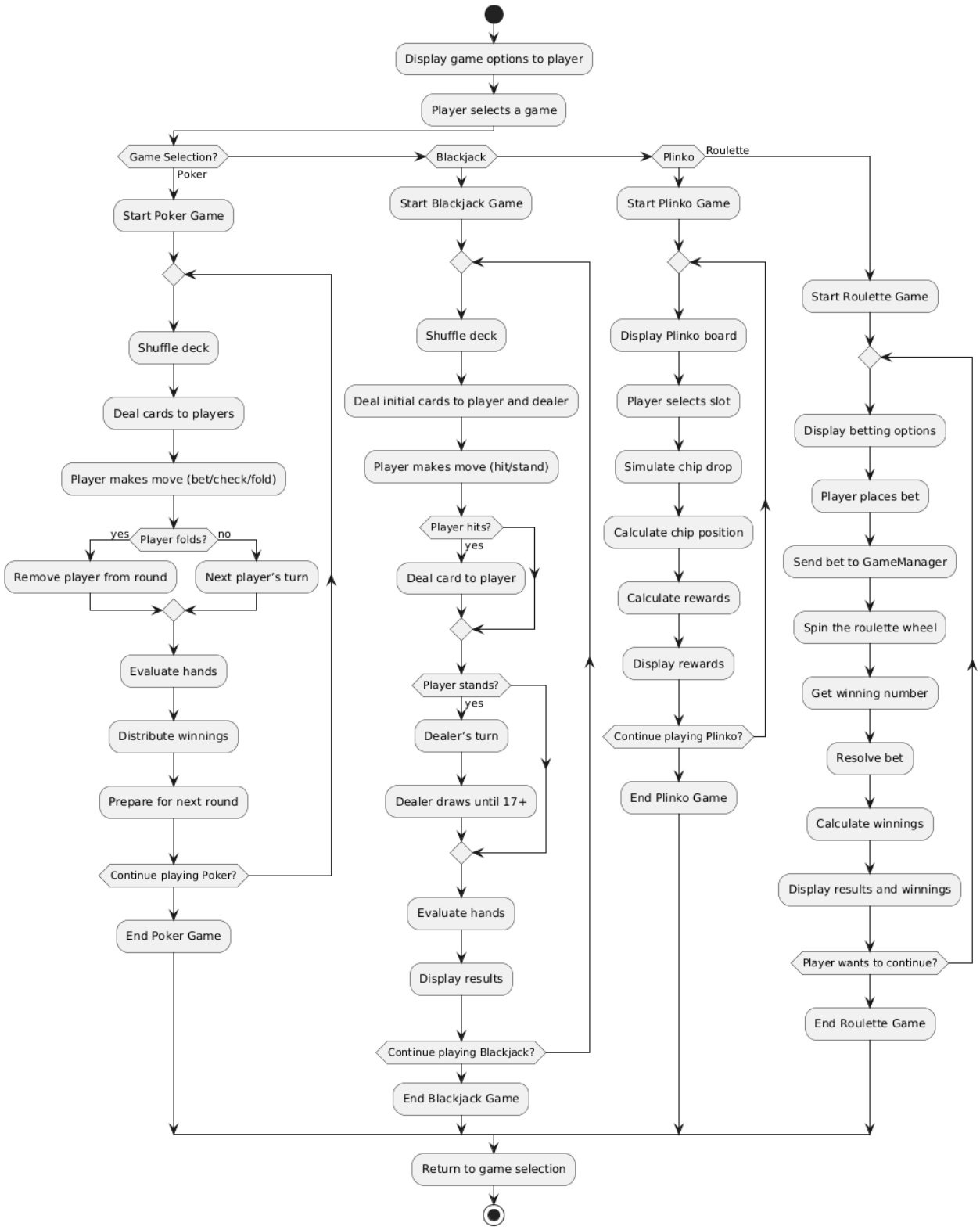
In the Plinko game, the player selects a slot for a chip to drop. The game simulates the chip falling through the pegs of the Plinko board, and based on where the chip lands, the player is awarded a reward. Afterward, the player can decide to play again or stop.

In Roulette, the player is shown betting options, places their bet, and the wheel is spun. The winning number is determined, and the player's bet is either resolved as a win or loss. The player’s winnings are calculated, and they can choose to keep playing or quit the game. In all the games, once a player finishes, they are brought back to the game selection screen where they can choose a new game or exit.

Login


Open the App
Display Login Screen
Player enters username and password
Send credentials to Firebase Authentication
Credentials Valid?
yes
no
Retrieve user data from Firebase Database
Load user data into client
Display user profile and home screen
Display login error
Prompt for re-entry of credentials

Action Diagram



Design Issues

Functional Issues

What do users need to create/sign into an account?

- Option 1: Username + Password
- Option 2: Email address
- Option 3: Phone number

Choice: Options 1 and 2
Justification: There should be a way to identify a user among other players (e. g. username). Additionally, there should be a way we can contact the user of news, updates, outages, account issues, etc. Hence, we will also ask for an email address.

What will we do in case a user leaves mid-game?

- Option 1: End the game for everyone
- Option 2: Replace the user with an AI bot
- Option 3: Completely remove the user, making the game one player less

Choice: Option 2
Justification: We believe that we should not compromise the game quality for the players that are still in the game; therefore, Options 1 and 3 do not align with our goal. Option 2 keeps the game going without randomly losing a player.

What happens if a user does not have enough currency to raise to the current bet?

- Option 1: Kick the user out of the game
- Option 2: Create a split pot so that the user can still raise
- Option 3: Make the user a spectator

Choice: Option 2
Justification: Once again, we hold paramount the fact that the user's experience should not be compromised. We will still allow the user to play even if they ran out of enough currency.

What if the users use profanity in chat or they threaten other players?

- Option 1: Ban the user immediately
- Option 2: Censor the profanity and ignore the threat
- Option 3: Create a way to report a user and once enough reports accumulate, the user may be banned

Choice: Option 3
Justification: To make the platform fair for everyone, we believe that one singular use of profanity should not be grounds for banning the user. Additionally, we cannot simply allow the user to indefinitely affect others' experience negatively. That is why we will keep track

of users' negative actions and, once enough of these actions accumulate, we will ban the user.

Non-Functional Issues

What database will we use for user details storage?

- Option 1: AWS S3
- Option 2: Proprietary SQL Database
- Option 3: Google Firebase

Choice: Option 3

Justification: AWS S3 offers high amounts of storage for any type of data; however, it is not a free platform. Coding our own SQL Database will require high amounts of effort and, additionally, we will have to find a server to host this database. Google Firebase offers a readily available database that suits our purposes and, furthermore, it can be used free of charge.

What model will we use for AI model training?

- Option 1: SKLearn Decision Trees
- Option 2: SKLearn Random Forest Regressor
- Option 3: TensorFlow Deep Learning Model
- Option 4: PyTorch Deep Learning Model

Choice: Options 1 and 3

Justification: Most gaming algorithms that exist are implemented using Decision Trees. However, this is a very simple model that may not suit the needs of poker. We will first test Option 1 and, if this does not suit our needs, we will go with Option 3. The reason for which we prefer Option 3 over 4 is that, although the two platforms are very similar, we have more experience with TensorFlow than PyTorch.

What is the main server going to be?

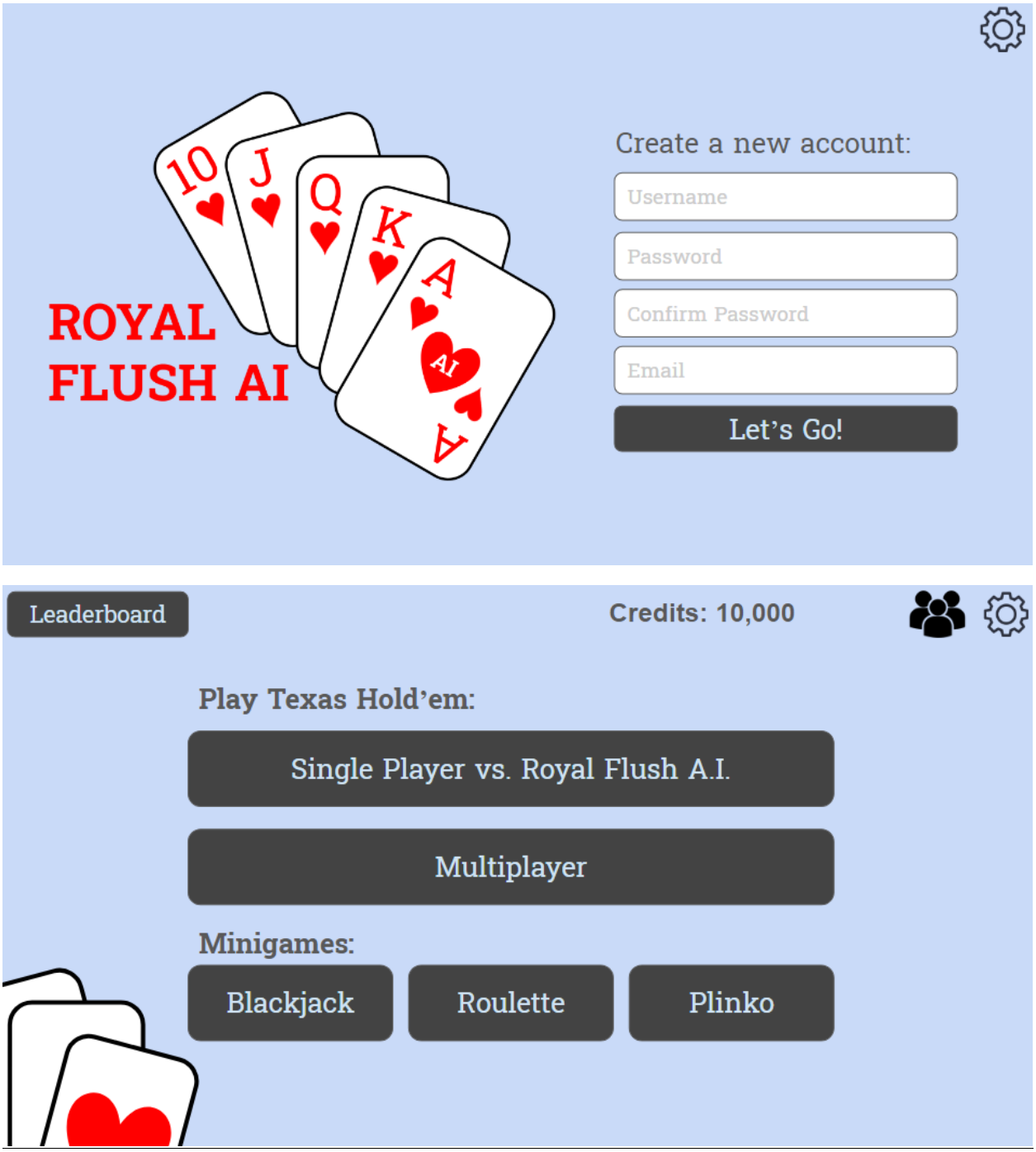
- Option 1: Proprietary Server
- Option 2: Photon Server
- Option 3: React server

Choice: Option 3

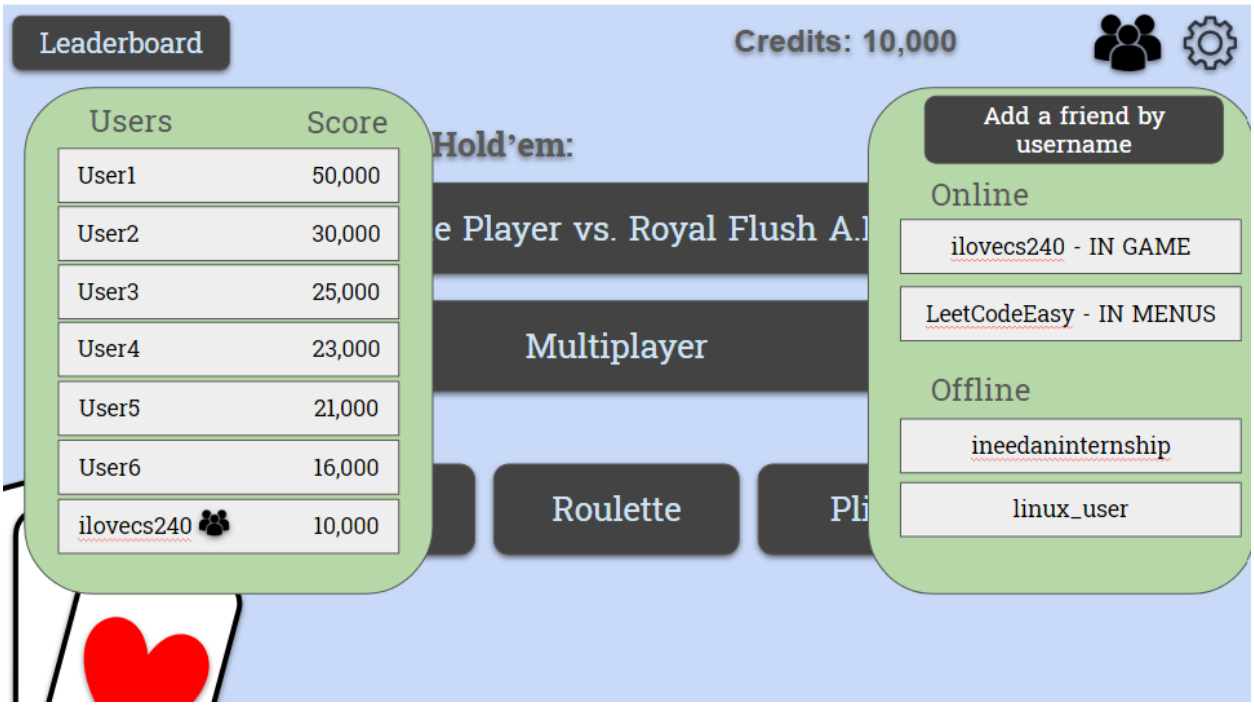
Justification: We feel that Option 1 is difficult to implement as we need to not only code a fully functional server, but we must also find a host for it. Option 2 is also a good choice; however, Photon server only work for certain gaming libraries that we will not user. However, we all have experience with React and there are many web hosting platforms designed for it.

UI Mockup

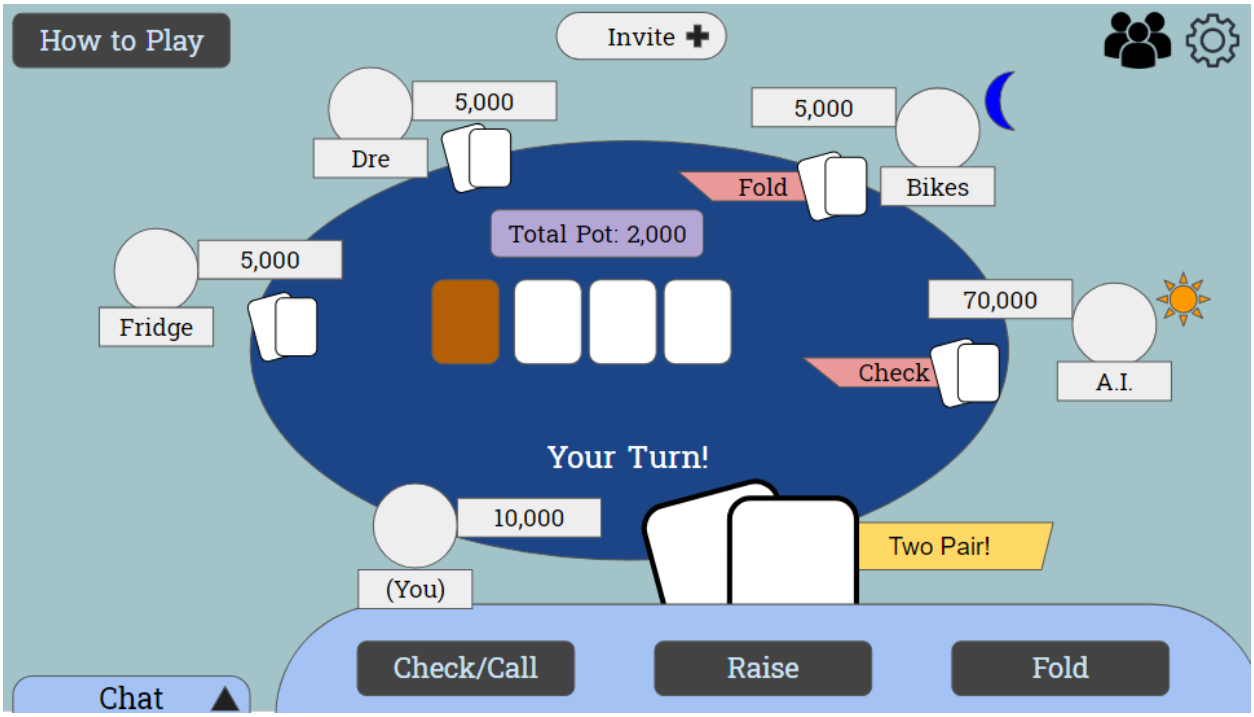
This UI Mockup displays the first screen that shows up when the application is opened. The settings button at this stage controls the appearance of the game and the volume of the game. New players would click the top button to create an account, and returning players would click the bottom button to sign into their account. This information is stored in Google's Firebase system.



This screen is what pops up after login and shows all the game modes available to play. There are a couple more buttons now, plus a more sophisticated menu. The settings button can now adjust profile appearance and security details. The icon directly to the left of settings will show a list of friends and their online status. Clicking the leaderboard will show the global leaderboard stored on cloud. The number of credits that the account has will be at the very top of the page. Of course, there are also going to be more sophisticated decorations for the background and the buttons.



Buttons like the leaderboard and the friends list will generate popups over the menu. The leaderboard displays the users with the highest scores globally. If one of them is your friend, an icon will be displayed next to them as well. When clicking the friends list, the users will be separated into online and offline sections. If there is a long list of friends, a scroll bar will appear. There is also a button that allows you to add a friend by their username. Clicking on one of your friends gives an option to spectate their game, chat with them, or remove them.



This interface is what the app looks like when a player is in a game. All the players with their usernames and avatars are displayed, with their customizations in place of the placeholder white circles. On the top left, there is a how-to-play menu with a guide on the game being played to guide new users. At the top is an invite button to invite other players to the game, as well as adding additional AI bots. There is a flair to represent big and small blind, as well as text that lights up when it is your turn to play. Important actions such as folding and checking will be highlighted after a player commits that action. If you have an advantageous hand, such as a pair, that will be highlighted to you and only you. The total pot and the number of credits each player has in game is displayed effectively to all players. On the bottom left is a chat that can be used to communicate with other players in game, and the other buttons on the bottom are used when it is your turn.

The web interface will update in real time for all players with the progression of the game.