# CA400 Functional Specification - KryptoKasino

## 0. Table of contents



## 1. Introduction

### 1.1 **Overview**

The system we will be developing is a web application which hosts traditional "Texas Hold'em" games of poker. Users can connect their cryptocurrency wallet to the web app and trade their cryptocurrency to enter games and place bets. In a game, it will be played online by at least two players with up to four players. When there are two players within a game lobby, a timer will begin to countdown when the game begins.

### 1.2 **Glossary**

Blockchain: A peer-to-peer network's entire transactions are recorded in a blockchain, which is a decentralised ledger.
Cryptocurrency: Digital currency.
Cryptocurrency Wallet: A tool to securely store and manage your cryptocurrency.
NFT: A non-fungible token which is unique and stored on the blockchain.
Web 3.0: Web3 is a proposed new iteration of the World Wide Web which would combine ideas like decentralisation, blockchain technology, and token-based economies.
Raise: In the same betting round, to raise means to increase the size of an existing wager.
Check: The action passes to the next player without us making any wager.
Fold: A player forfeits.

## 2. General Description

### 2.1 Product / System Functions

The user will enter the home page and have 2 options. One option will be to make an account with Krypto Kasino. This will involve entering their username, email address and password. They will then be prompted to link their crypto wallet with the system which will be matched with their account on the system.

The user will also have the option to not make an account with Krypto Kasino and instead just link their crypto wallet with the system. If they choose to do that then when entering and playing poker games on the system they will just be a guest. This will be reflected in their username when playing games, ie. GuesUser-XXXX where X is a random number assigned to them when they're playing to distinguish them from other guest players.

From here, the user will have the option to deposit funds from their crypto wallet into their account with Krypto Kasino. The user will be able to add this money back into their crypto wallet at any time when on the home page.

In order to improve the user experience of the system, when a user deposits cryptocurrencies from their wallet into their system account it will be displayed to them in its crypto amount as well as the EURO amount. This is to help users make sense of how much funds they have in their account.

Once the user has deposited funds from their crypto wallet into their account, they will have the option to enter public poker games. They will click a ‘play a game’ button which will put them onto a loading screen. The system will then search for public games already in action that have space for another player. If there is a game with space, the user will be prompted to select the amount of crypto currency they would like to enter the poker game with and then enter and start playing. If there are no games currently being played or there are no games with space, then the user will be launched into a new game where the system will scan for other users looking to enter a game. Anytime a user enters a game, they will be prompted to select the amount of crypto they wish to play with in the game in a EURO amount.

The user will then play poker with their crypto against other players with the option of leaving the game whenever they want. When a player leaves the game, the amount of money they have will be then updated to their system account where it will be displayed on the main dashboard in its updated crypto and EURO amount.

### 2.2 User Characteristics and Objectives

The user community for our crypto poker game will own cryptocurrencies and have interest in the game of poker. Due to the fact that users of this system would need to be holding cryptocurrencies in a crypto wallet, we can assume that many of the users would be familiar with linking their crypto wallets to websites in order to make use of their crypto. Web3.0 apps and websites all require that users link their crypto wallets before making use of them. The user community who will be using our system is a subset of this cryptocurrency / web3.0 market, that is owners of cryptocurrencies who also like playing poker. Therefore it’s important that linking your crypto wallet to the website is a seamless process.

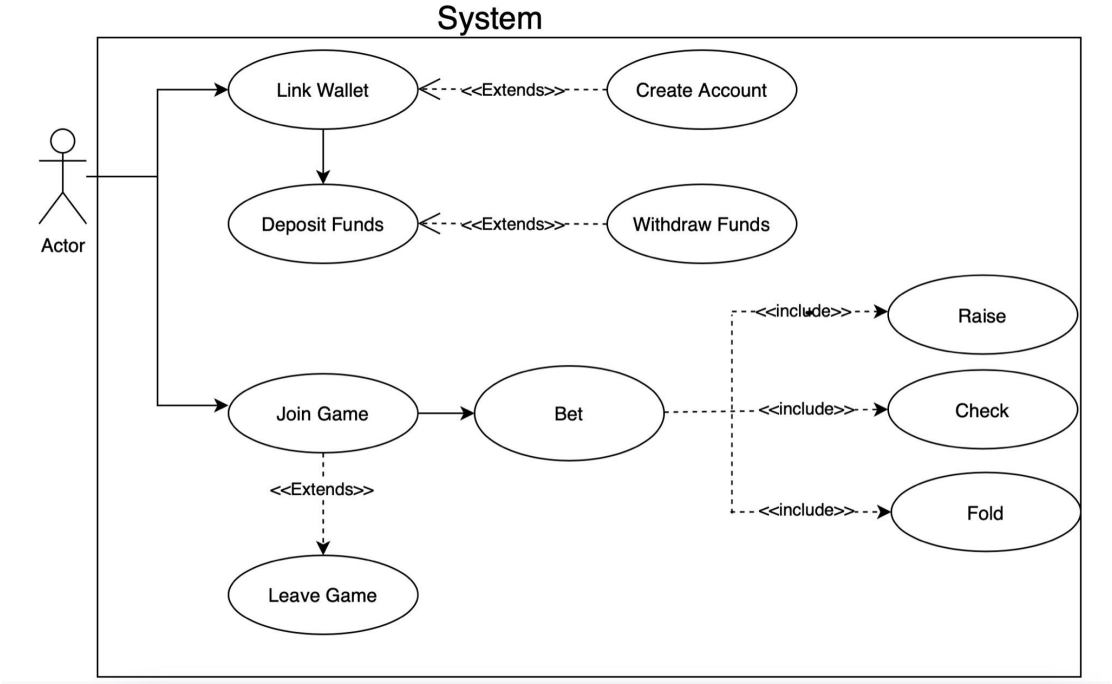
Furthermore, it’s fair to say that a lot of the users who would be using our system will have prior experience in online poker, whether it is for real money or just online poker games that don’t use real money. Online poker is very popular today and Web3.0 poker is really just leveraging this already existing online poker market into a new one. However that doesn’t mean that all users would be familiar with playing online poker. It is then imperative that the experience of joining and playing games be simple and easy to do.

Overall, our goal with this software system is to create an environment that is easy to use and follow along with. Despite the fact that the user community of our software system may, for the most part, be familiar with linking crypto wallets with online systems and playing online poker, we still want to make the processes simple. There are existing systems out there that do the same thing as ours, so a way for ours to stand out would be to create a UI/UX that is better than the rest. Yes, we want a fully functioning multiplayer poker game that facilitates blockchain transactions for users to make cryptocurrencies, but doing so in a way that is easy and fun for the users is the main objective.

Under the context of a ‘wishlist’ for our crypto poker game, something that we could look to integrate down the line would be the use of NFTs which stands for Non-Fungible Token. An NFT is basically a digital item that cannot be replaced, it is one of a kind. These could be leveraged as a rewards system for the users of our game. If a player wins a game she can receive an NFT that gives her special perks when using our system. As was mentioned earlier, the overall goal of the system is to create a fully functioning multiplayer crypto poker game that is quick, easy and fun to use for the users. But the use of NFTs would be a nice addition to add to that down the line.

## 2.3 Operational Scenarios

### 2.3.1 Use Case Diagram




### 2.3.2 Operational Scenario 1

Bob creates an account with Krypto Kasino entering his unique username, unique email address and valid password. He links his MetaMask wallet with the system which will be attached to his account. He then deposits 0.01 BTC from his cryptocurrency wallet into his Krypto Kasino account. His account balance displays as 197.50 EURO. He enters a public game with a stake of 0.05 BTC. Bob plays Poker against other players. Bob loses all of his stake. He leaves the game. His account balance is updated to 98.29 EURO.

### 2.3.3 Operational Scenario 2

Alice links her cryptocurrency wallet with the system. She does not create an account therefore she is assigned the username “GuesUser-1” instead. She enters a

public game with 0.2 ETH. Alice wins the pot of her game. As Alice entered the game as a guest user, all winning funds 300 EURO are converted to ETH and are deposited into her wallet by the system. If Alice wishes to play another game, she must make another deposit transaction from her wallet.

## 3. Functional Requirements

### 3.1 Link Wallet

| | | |
|---|---|---|
| **Use Case 1** | | Link Wallet |
| **Criticality** | | High |
| **Goal in Context** | | Player link their cryptocurrency wallet to the system. |
| **Scope & Level** | | System, core. |
| **Preconditions** | | Wallet is unique in the current pool of players. |
| **Success End Condition** | | Player can now deposit funds to join games. |
| **Failed End Condition** | | Player cannot continue to join games. |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | Player selects to link their wallet. |
| **Description** | **Step** | |
| | **1** | Player selects to link their cryptocurrency wallet to the system. |
| | **2** | Player is prompted to enter their wallet address. |
| | **3** | Wallet uniqueness is checked. |
| | **4** | A link is created between the player and the wallet. |
| **Extensions** | **Step** | **Branching Action** |
| | **2a** | Wallet address does not exist: Prompt - reenter new address |
| | **2b** | Wallet address exists |
| | **3a** | Wallet address is already in use - Prompt - Wallet address is already being used |
| | **3b** | Wallet address is unique |

### 3.2 Create Account

| | | |
|---|---|---|
| **Use Case 1** | | Create Account |
| **Criticality** | | Medium |
| **Goal in Context** | | Player creates an account. |
| **Scope & Level** | | System, core. |
| **Preconditions** | | Player has selected the option to create an account |
| **Success End Condition** | | Player have a unique account on Krypto Kasino |
| **Failed End Condition** | | Player play as a guest |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | Player selects to button to create an account |
| **Description** | **Step** | |
| | **1** | Player selects a button to create an account on the system. |
| | **2** | Player is prompted to enter a unique username consisting of at least five characters, a valid email and a valid password. |
| | **3** | Unique account is created. |
| **Extensions** | **Step** | **Branching Action** |
| | **2a** | Username already exists: Prompt - Enter another username. |
| | **2b** | Username too short: Prompt - Username is not valid. |
| | **2c** | Username is unique. |
| | **2d** | Email address is already in use - Prompt - Enter another email address |
| | **2e** | Email address is unique |
| | **2f** | Password is too short: Prompt - Password is not valid. |
| | **2g** | Password is valid |

### 3.3 Deposit Funds

| | | |
|---|---|---|
| **Use Case 1** | | Deposit Funds |
| **Criticality** | | High |
| **Goal in Context** | | Player deposit funds from their cryptocurrency wallet into their account. |
| **Scope & Level** | | System, core. |
| **Preconditions** | | 1. A wallet is linked to their account.<br>2. Funds exist within their wallet. |
| **Success End Condition** | | Player can continue to join games. |
| **Failed End Condition** | | Player cannot continue to join games. |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | 1. Player selects the button to deposit funds. |
| **Description** | **Step** | |
| | **1** | Player selects the button to deposit funds. |
| | **2** | Player is prompted to enter a deposit amount. |
| | **3** | Wallet is checked for the selected amount. |
| | **4** | The funds are moved from their wallet to their account. |
| **Extensions** | **Step** | **Branching Action** |
| | **2a** | Deposit amount is lower than the minimum deposit: Prompt - Enter a higher amount |
| | **2b** | Deposit amount is greater than the minimum amount required |
| | **3a** | Wallet does not contain the selected amount of funds: Prompt: Transaction error. |
| | **3b** | Wallet contains the selected amount. |

### 3.4 Withdraw Funds

| | | |
|---|---|---|
| **Use Case 1** | | Withdraw Funds |
| **Criticality** | | High |
| **Goal in Context** | | Player withdraw funds from their account into their cryptocurrency wallet. |
| **Scope & Level** | | System, core. |
| **Preconditions** | | Funds exist within their account. |
| **Success End Condition** | | The funds are withdrawn from their account to their cryptocurrency wallet. |
| **Failed End Condition** | | Player cannot withdraw funds. |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | Player leaves the game as a guest or Player selects the button to withdraw funds as an account holder |
| **Description** | **Step** | |
| | **1** | Player selects the button to withdraw funds as an account holder |
| | **2** | Player is prompted to enter a withdrawal amount. |
| | **3** | Wallet is checked for the selected amount. |
| | **4** | The funds are moved from their account to their wallet. |
| **Extensions** | **Step** | **Branching Action** |
| | **1a** | Player leave game as a guest |
| | **2a** | Player leave the game as a guest therefore all funds are withdrawn. |
| | **2b** | Wallet does not contain the selected amount of funds: Prompt: Transaction error. |
| | **2c** | Wallet contains the selected amount. |
| **Variation** | **Step** | **Branching Action** |
| | **1a** | Player leaves the game as a gues**t** |
| | **2a** | Player leaves the game as a guest therefore all funds are withdrawn. |

### 3.5 Join Game

| | | |
|---|---|---|
| **Use Case 1** | | Join Game |
| **Criticality** | | High |
| **Goal in Context** | | Player joins a table. |
| **Scope & Level** | | System, core. |
| **Preconditions** | | 1. Server has spare capacity<br>2. User has navigated to the homepage. |
| **Success End Condition** | | The player has been granted a place within an ongoing game or has joined a newly created lobby and waits for the game to start. |
| **Failed End Condition** | | Player cannot join a table. |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | Player selects to join a table. |
| **Description** | **Step** | |
| | **1** | Player selects to join a game. |
| | **2** | Player is prompted to enter a stake. |
| | **3** | Table availability is checked |
| | **4** | Player is inserted randomly into any available seat at the table. |
| | **5** | Game assigns chip amount in accordance to their stake and turn order is established. |
| | **6** | Timer to join the table begins (60s). If player > 1, vote is taken to start the game or wait for the timer. |
| | **7** | Game starts. Game distributes 2 random cards to each player and randomly selects 5 community cards. |
| **Extensions** | **Step** | **Branching Action** |
| | **2a** | The stake chosen is not sufficient to join this game:<br>Prompt: Select a higher stake. |

| | | |
|---|---|---|
| | **3a** | No tables open & server capacity filled: prompt – try again later. |
| | **3b** | No tables open but capacity available: new table created. |
| | **3c** | Open table with 1 or more spaces available. |
| | **6a** | Timer elapses, no player joins and only 1 player is present: timer resets. |
| | **6b** | New player joins table in time: Return to and continue from step 5. |
| | **6c** | Majority votes to start: Continue to step 7. |
| | **6d** | Majority votes to wait: Timer continues. |

**3.6 Leave Game**

| | | |
|---|---|---|
| **Use Case 1** | | Leave Game |
| **Criticality** | | High |
| **Goal in Context** | | Player leaves a table. |
| **Scope & Level** | | System, Core. |
| **Preconditions** | | The player is currently in a game. |
| **Success End Condition** | | The player is no longer in a game. |
| **Failed End Condition** | | Player still exists in the game. |
| **Primary, Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | The player presses the “Leave Game” button. |
| **Description** | **Step** | |
| | **1** | Player selects the option to leave the game. |
| | **2** | System removes player from the game. |
| | **3** | The game refreshes the turn and seating order. |

| | | |
|---|---|---|
| | **4** | Player is returned to the homepage. |
| **Extensions** | **Step** | **Branching Action** |
| | **1a** | The player has taken part in the current hand: Pot remains unchanged. |
| | **1b** | The player has already folded. |
| | **1c** | The game has not yet begun: timer resets. |

### 3.7 Check

| | | |
|---|---|---|
| **Use Case 1** | | Player Checks |
| **Criticality** | | High |
| **Goal in Context** | | Player passes the next action onto another player at the table. |
| **Scope & Level** | | System, Core. |
| **Preconditions** | | 1. No other player raised during the current round<br>2. User has not folded during the current hand<br>3. User has already matched the highest bet |
| **Success End Condition** | | Action is passed onto the next player. |
| **Failed End Condition** | | Action is not passed onto the next player. |
| **Primary,<br>Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | The player's turn has begun. |
| **Description** | **Step** | |
| | **1** | Game determines if preconditions have been met. |
| | **2** | Action is passed onto user and 30 second timer starts for their turn |
| | **3** | User elects to check. |
| | **4** | Action is passed onto the next player in the current round. |
| **Extensions** | **Step** | **Branching Action** |

| | | |
|---|---|---|
| | **1a** | Preconditions have failed: Check option is unavailable |
| | **1b** | Preconditions checked and passed. |
| | **3a** | User doesn't select an action before 30 seconds runs out: Automatic check and action passed onto next player |

**3.8 Raise**

| | | |
|---|---|---|
| **Use Case 1** | | Player Raises |
| **Criticality** | | High |
| **Goal in Context** | | User bets their chips during the current round of betting |
| **Scope & Level** | | System, Core. |
| **Preconditions** | | User has adequate amount of chips to place their bet |
| **Success End Condition** | | User chips are added to the pot for the current hand.<br>Action passed onto next player |
| **Failed End Condition** | | User doesn't have adequate amount of chips to play bet |
| **Primary, Secondary Actor(s),** | | Player,<br>System |
| **Trigger** | | Action is passed onto the user during the current round |
| **Description** | **Step**<br>**1** | Action is passed onto user and 30 second timer starts for their turn |
| | **2** | User elects to raise before 30 seconds runs out |

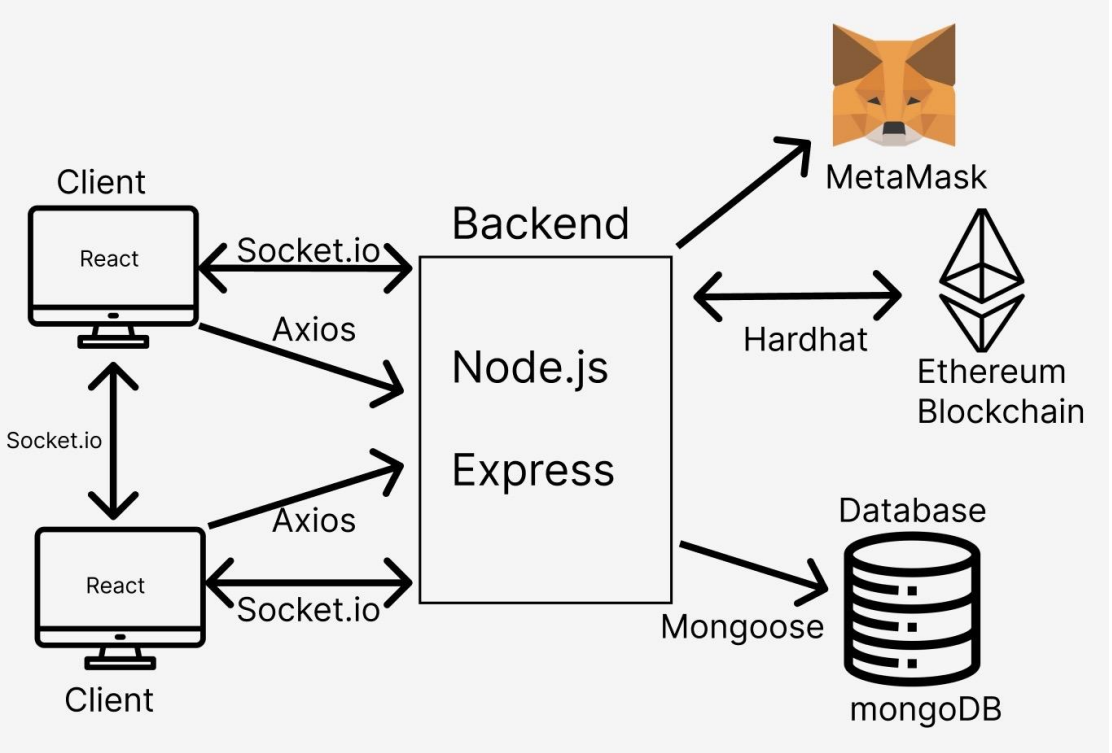
| | | |
|---|---|---|
| | **3** | A text box prompt comes up on screen and user can select how many chips they want to bet |
| | **4** | Chips are added to the pot and action is passed onto next player in the hand |
| **Extensions** | **Step**<br>**2a** | **Branching Action**<br>User doesn't select an action before 30 seconds runs out: Automatic check/fold and action passed onto next player |
| | **3a** | User doesn't have enough chips to place bet they entered into the text box: Still the user's go until 30 seconds runs out |
| | **3b** | User can select a max bet meaning they elect to put all their chips into the pot |

**3.7 Fold**

| | |
|---|---|
| **Use Case 1** | Player Folds |
| **Criticality** | High |
| **Goal in Context** | User folds their cards to give them up and sit out on the current round |
| **Scope & Level** | System, Core. |
| **Preconditions** | 1.The player is active<br>2. A previous player chose to raise during the round of betting |
| **Success End Condition** | The player loses their cards and is marked inactive<br>Action is passed onto the next player |
| **Failed End Condition** | Player is still active during the round |
| **Primary,** | System |

| | | |
|---|---|---|
| **Secondary Actor(s),** | | |
| **Trigger** | | Action is passed onto the user during the current round |
| **Description** | **Step** | |
| | **1** | Action is passed onto user and 30 second timer starts for their turn |
| | **2** | User elects to fold before 30 seconds runs out |
| | **3** | Game takes cards away from the user |
| | **4** | User is marked inactive |
| | **5** | Action is passed onto next player in the current round |
| **EXTENSIONS** | **Step** | **Branching Action** |
| | **2b** | User doesn't select an action before 30 seconds runs out: Automatic check/fold and action passed onto next player |
| **VARIATIONS** | | **Branching Action** |
| | **1** | User is the last active player: user wins |

## 4. System Architecture




- React: Users interact with the system through an interactive React user interface.
- Socket.io: This allows web clients and servers to communicate in real time and in both directions. It comprises a server-side library for Node.js and a client-side library that runs in the browser.
- Axios: Handle's HTTP requests and communicates with the backend.
- Node.js: Server-side programming for the website and back-end API service.
- Express: Assists Node.js in helping manage the server and routes.
- MongoDB: Document database for storing user's account information.
- MetaMask: Facilitates users to connect their cryptocurrency wallet to the system.
- Hardhat: Allows the system to interact with the Ethereum Blockahin.
- Ethereum Blockchain: Enables the writing of smart contracts in accordance to the users' interaction with the system.

## 5. High-Level Design

Building a cryptocurrency based poker web application has a lot of parts that go into it. There are many components involved that must be able to contact each other to pass data around. The better the system can do this, the more efficiently it will run.

The client side of the web application will contain a react application. The client will contain all of the frontend code for the system which will be written with ReactJS. All of the possible web pages that a user will be able to see / interact with will be written up here. This will start off with just a main home

page where the user will be able to create an account with our system. Proceeding this way will prompt the user to create an account by filling in a username, email and password followed by linking their crypto wallet to the system as well. If they don't have a crypto wallet they will be redirected to an external web page that will allow them to do so.

These frontend web pages that will make up our web app will also be responsible for importing our smart contracts. Acquiring the necessary data, such as, the contract Address, ABI and Signer will allow the facilitation of on-chain transactions for the poker game that will be integrated into the website.

Alternatively, the user can choose to only link their crypto wallet to the website without actually creating an account. If they proceed this way, this will mean that they are proceeding as a guest.

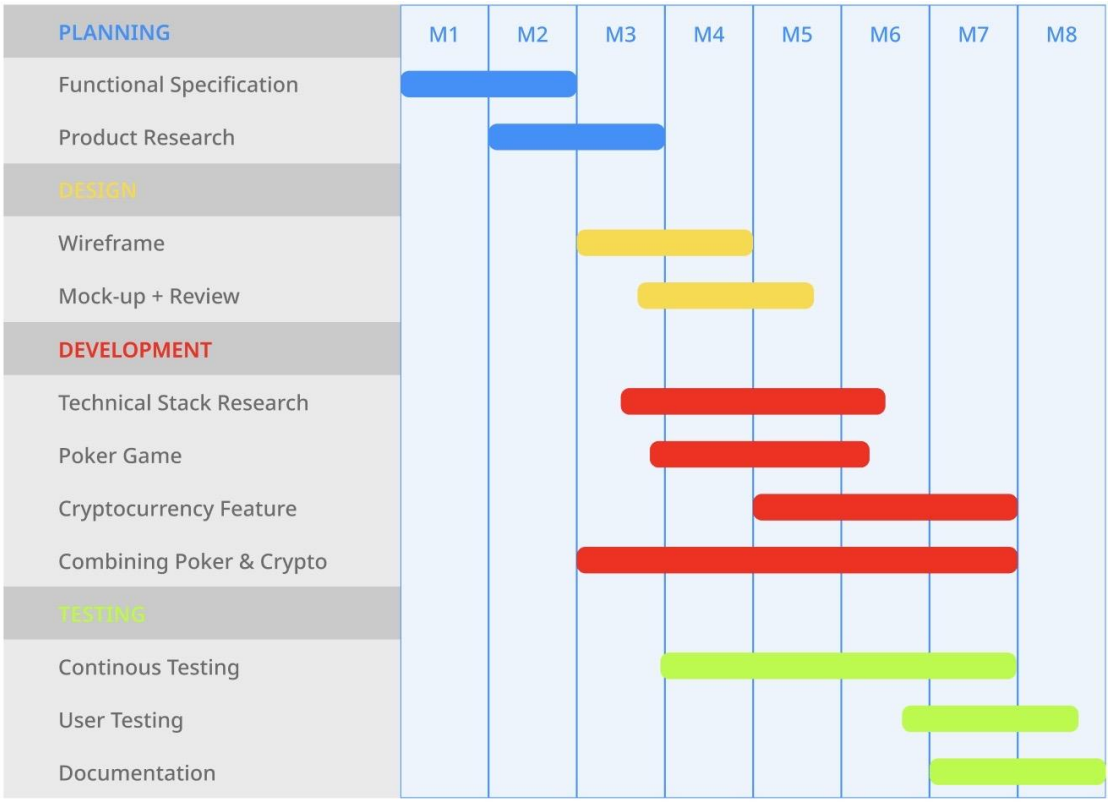
Our poker web application's backend portion will contain all of the source code for our poker logic. The classes will include information about low-level poker logic. Here is where the main logic for the game of poker will be built up such as the structure for a card, the deck of cards, the hands that can be played in a game of poker etc.

On top of this, the backend of the system will also make use of websockets to facilitate the multiplayer aspect of our poker web app. We will use the SocketIO library to attach our websocket engine to the system server. This server data will then be fed to our React app on the frontend.

Our smart contracts will be written in Solidity under our smart contract folder alongside the client folder. Our React app will then interact with these smart contracts. This is also where users linking their crypto wallet will be integrated into the code. The user will enter the website and there will be a button to press for them to link their crypto wallet with the website. If they are revisiting the website and have already linked their crypto wallet to the site, there will be no button displayed for them to link them.

# 6. Preliminary Schedule

### 6.1 GAANT Chart




### 6.2 Project Timeline

| Timeline | Tasks |
|---|---|
| Sept | • Brainstorm Project Ideas<br>• Project Research |
| Oct | • Draft Proposal Form<br>• Find Supervisor<br>• Purpose Idea |
| Nov | • Functional Specification<br>• Plan Design<br>• Wireframe<br>• Mock-up & Review |
| Dec-Jan | • Develop Web App<br>• Poker Logic<br>• Sockets<br>• Continuous Testing |
| Feb-Mar | • Backend<br>• Smart Contracts<br>• UI/UX Implementation |

| | |
|---|---|
| April | • User Testing<br>• Documentation |