Software Requirements Specification

WebGL Card Game Platform

Version 1.3

3/15/19

Team WebGL Game

https://docs.google.com/document/d/1of-L2X5ThC4Foo_qNy0bsRURRdHwSEwFtL7rQPF1dT8/edit

Bellevue College

Sara Farag

Revisions

Version	Primary Author	Description of Version	Date completed
1.0	Sean Hardin, Anthony Klobas, Jeffrey Talada	Created primary version of document	10/17/18
1.1	Jeffrey Talada, Sean Hardin	Finalized for Milestone 1	12/4/18
1.2	Anthony Klobas	Updated Requirements	1/24/19
1.3	Anthony Klobas, Sean Hardin, Jeffrey Talada	Updated client functions, database currently implemented, and removed User Characteristics	3/15/19

Table of Contents

1. Introduction

1.1 Purpose

This document specifies and lays out the assumptions, constraints, and requirements of a WebGL based card game platform.

This document is intended to be used as a reference by Dr. Sara Farag, the project developers, and potentially future developers.

1.2 Scope

The scope of this system is to build an online card gaming platform that utilizes WebGL graphics. It will support hosting of secure games with multiple players including AI players. The system shall support running multiple games at a time on the server side. The system will include at least two games (Blackjack and Texas Hold'em) and include a betting system. A player may login to maintain a win/loss and balance history. The platform shall only support turn-based card games. The system won't support saving games. If there is time, other functionalities like chat, more AI options, and more playable games will be included.

The benefits of building a platform as opposed to a single game include extensibility in security, support for the basic suite of functionalities (multiplayer, AI, chat, betting), and easy development of new card games.

1.3 Definitions, Acronyms & Abbreviations

- AGILE - an approach to software development that emphasizes "adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change." <https://en.wikipedia.org/wiki/Agile_software_development>
- AI - artificial intelligence
- Client - The application running inside of a user's web browser that communicates with our server to run the game
- DOM - Document object model
- ES6 - A more recent version of javascript with a module system
- FPS - Frames per second
- Fragment - Small section of a polygon to be rendered (think pixel)
- GLSL - OpenGL Shading Language
- GPU - Graphics processing unit
- IDE - Integrated Development Environment
- JS - Javascript
- JSON - Javascript Object Notation
- Player - A person using the client to access the system
- RDBMS - Relational Database Management System
- UI - User Interface

- User - A person using the client to access the system
- Vertex - a point at a "corner" of a polygon
- Server - The application running the website and validation code for game play
- Shader - a program written to transform and display an object
- Z depth - the distance an object is from the observer, used for occlusion

1.4 References

Response Time Limits - nngroup.com, ‘Response Times: The 3 Important Limits’, 1993. [Article]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/. [Accessed: 4- Dec- 2018].

1.5 Overview

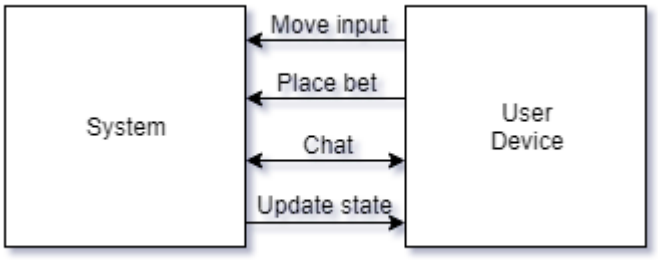
Section 2 of this document provides an overview of the system shown through the use of various diagrams along with simple descriptions of expectations for it. We begin with interfaces that the system will interact with, followed by a summarized list of the functionality that will be implemented, and potential users who will use the system. Lastly, it lists off constraints and assumptions that limit the system. Section 3 contains all the requirements of the game platform fleshed out in detail, beginning with the requirements needed to make the system run at all, then explaining the requirements which will improve user experience.

2. Overall Description

2.1 Product Perspective

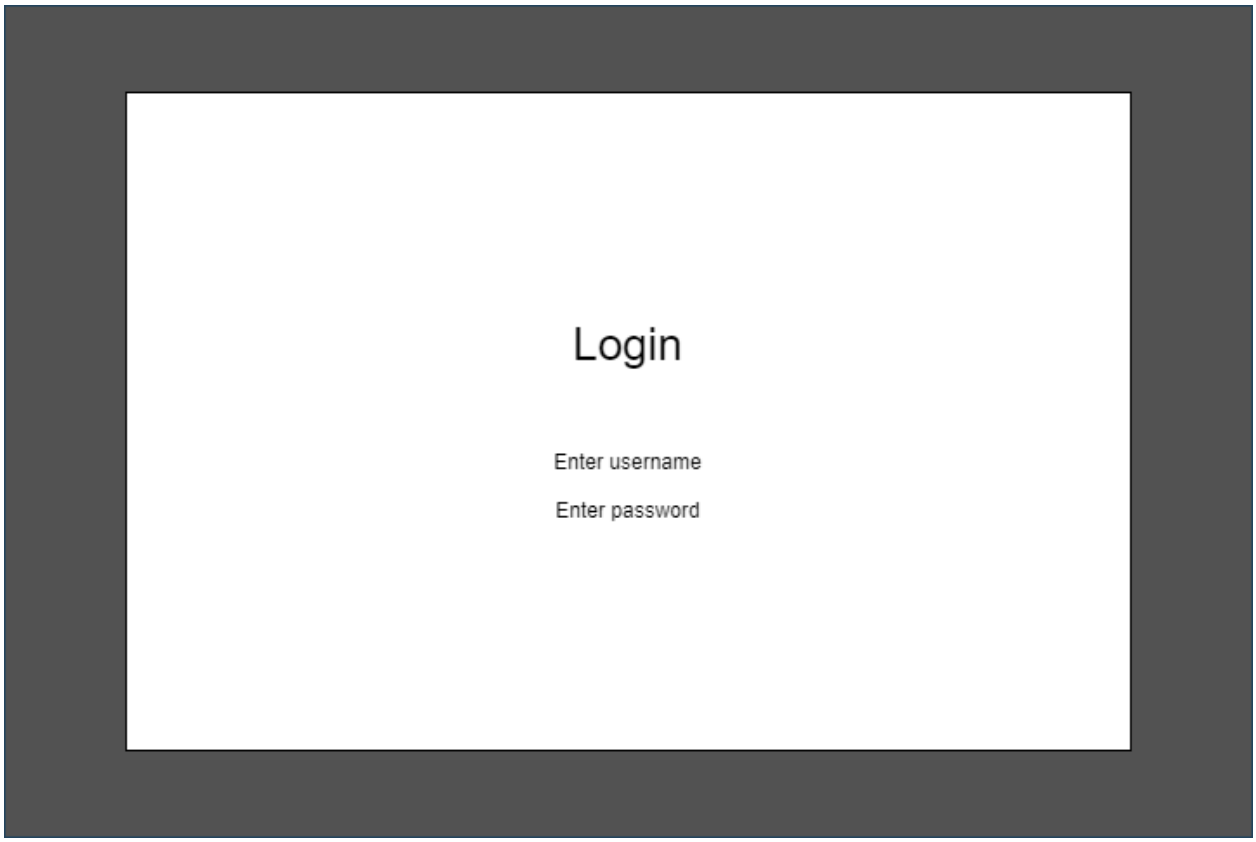
The system is based on the client-server model. Upon visiting the website, users will load a copy of the client which connects them to the game server. The user interacts with the system through the client UI which will exchange move, bet, and chat data with the server over the internet to manipulate the game's state. The system is dependent upon the user's device supporting WebGL.

Context Diagram



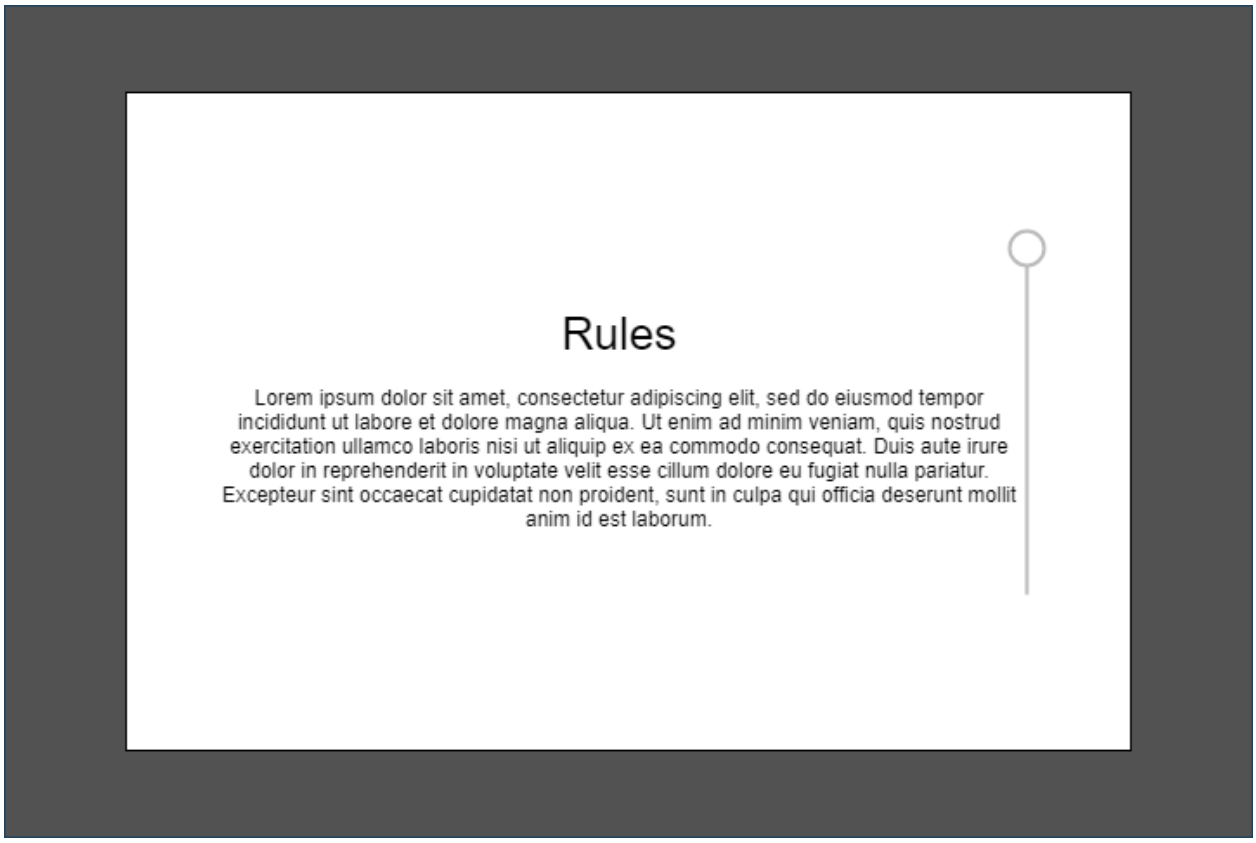
2.1.1 User interfaces

Login Screen



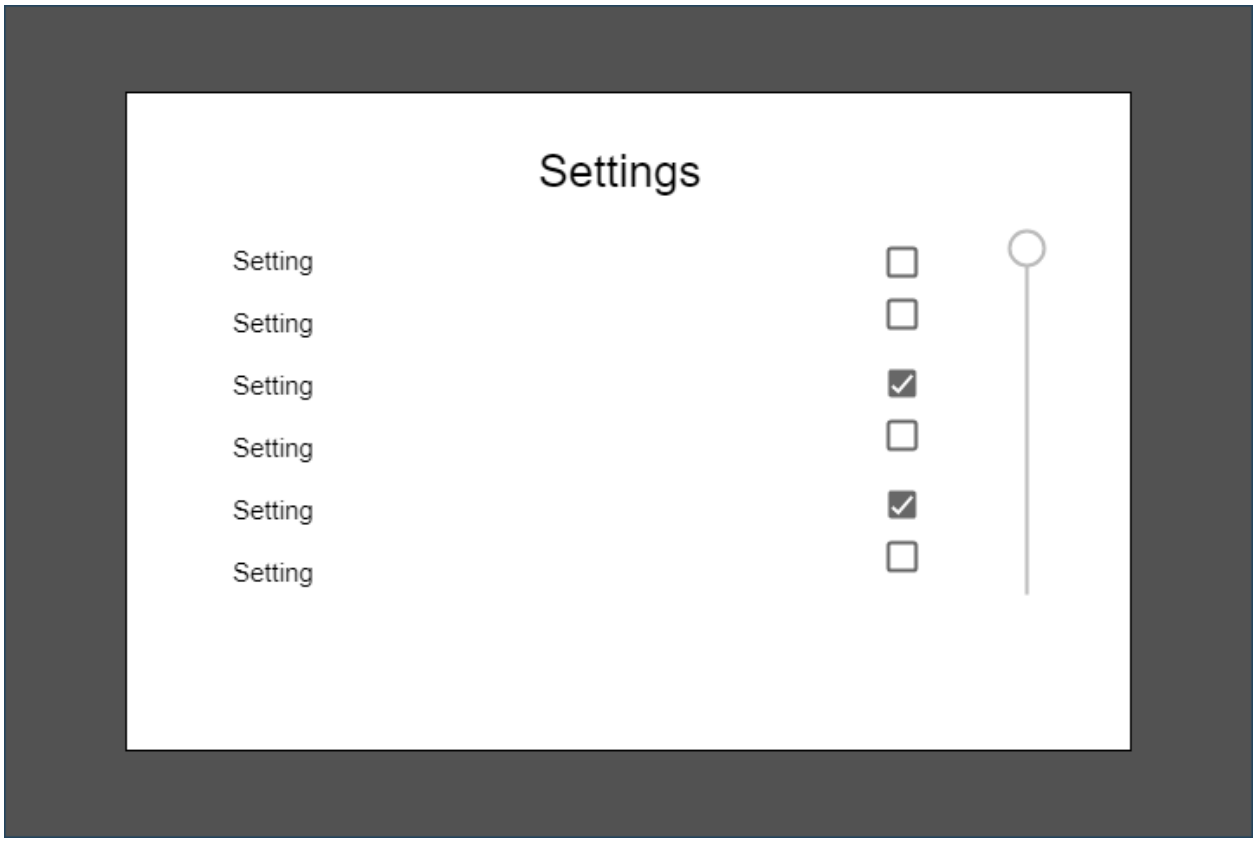
Users will be greeted with a login screen that asks for a username and password.

Rules Screen (Not yet implemented)



If a user is unfamiliar with a game's rules, the user can access them at any time during game play. If the rules are longer than a page, the user will be able to scroll through them. Clicking in the grey area will return the user to the game play screen.

Settings Screen

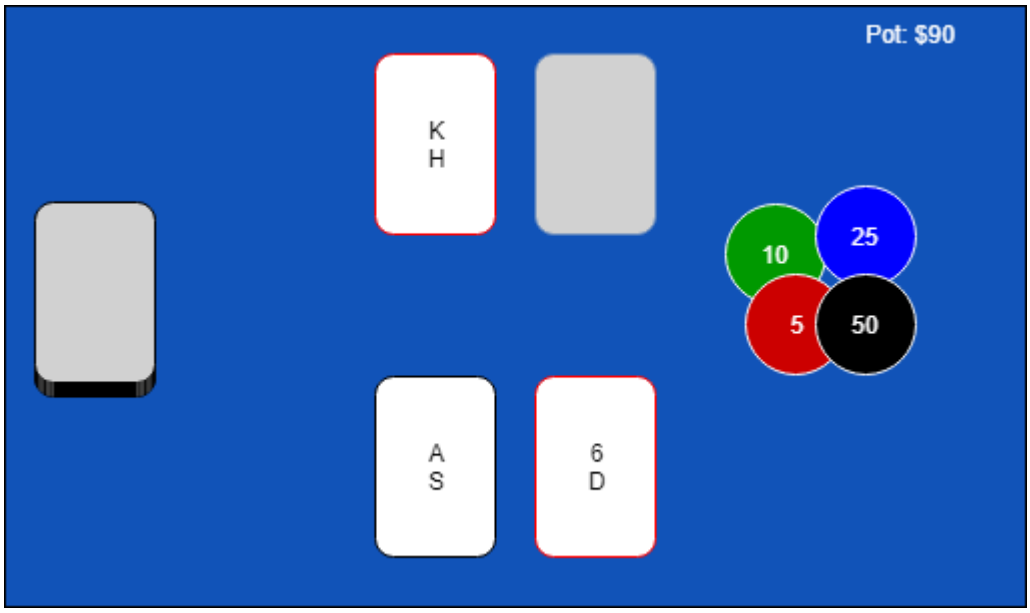


Should a user be interested in non-default game play, they can alter the game rules through the settings screen by checking and unchecking boxes.

Game Play Layout

Game Play Area	Player and other player information display (Banks, bets, current cards)
Chat	Settings, Rules, Game play buttons

Game Play Area

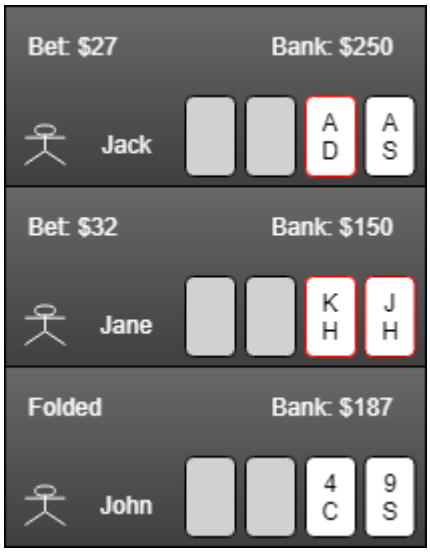


Depending on the specific game, the table layout could be drastically different. The current pot may be shown on the table represented as chips and as text. The player's current hand is displayed at the bottom of the game play area. Other visible hands may also be displayed. Cards may be in stacks, singles, or spreads. The player can click and drag cards around and they will snap into place if near a certain location as long as the move is legal.

Chat Area

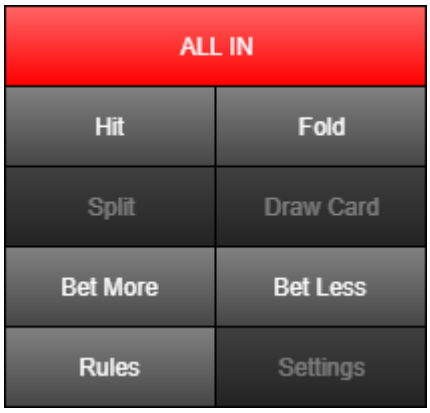
The chat interface will allow players to talk to each other while sharing a table. "All" will show everything from all tabs. "Chat" messages are visible to all players. "Whisper" will allow players to talk to a specific other player. "Console" will display messages related to trying to bet more than the player has or making illegal moves, turn taking, and game state updates.

Player Information Area



The player information area contains a summary of all the player's states. It shows their current bets, banks, and hands. The number of players will depend on which game is being played and how many people joined. The maximum and minimum number of players for a game will be specified in individual game rules.

Settings, Rules, Game Play Button Area



The button area will change drastically depending on the specific game being played. It should hold buttons relevant to the game being played as well as rules and settings buttons. Inactivated buttons will be darkened. Buttons may have different colors for emphasis. Other than clicking to select cards, this will be the primary way the player engages in game play.

2.1.2 Software interfaces

- WebGL
- Database (Optional)

2.1.3 Communication Interfaces

- TCP-IP
- SSL (Optional)

2.2 Product Functions

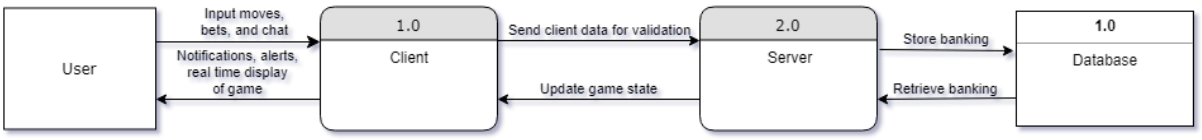
2.2.1 Client Functions

- Client connects to server
- Client page loads buttons for every available game option
- Client can select a type of game to join
    - Appropriate settings screen appears when choice is made
- Client can modify the game settings to be used in their game
- Client can request to join an existing game session
- Dynamically load in buttons based on the game chosen
- Dynamically load in game model based on choice
- User moves sent to server
- Placed bets sent to server
- Chat sent to server
- Notifies user when making an invalid move
- Display game on webGL canvas
- Display other player information in top right block of page as gamestate is received
- Display chat in the chatbox on the bottom left of the page

2.2.2 Server Functions

- Listen for user connections
    - On connection listen for user requests
- Create lobby for each type of game being hosted
- Create new game sessions for player
- Support multiplayer games
- Add players to existing game sessions
- Remove players from games when they lose connection and reorganize data to let games continue
- Forwards chat to all players in their own sessions
- Check that player moves are allowed based on the current gamestate
    - Block and notify the player if not allowed
    - Apply the move to the game model if it is allowed
- Send updated gamestate to players in a session
    - Hide the parts of the gamestate that individual players should not see
- Remove sessions that have no players in them
- Apply specific game logic as is written in in the individual game files.

Data Flow Diagram



2.3 Constraints

- Syncing clients and server across the internet
- Screen size/resolution
- Fat fingers
- Users attempting to cheat
- Database to server communication (optional)
- Internet connection speed
- Future game development

2.4 Assumptions and Dependencies

- Users have an up to date version of their preferred browser that supports WebGL and ES6 Modules
    - Edge 16+
    - Firefox 60+
    - Chrome 61+
    - Safari 11+
- Users have a device capable of running WebGL
    - The user's GPU must support OpenGL ES 2
    - Failing that, the user's processor must be fast enough to handle gpu calls in emulation
- Users have at least a 56kbps internet connection
- Users CANNOT have javascript blocked in their browser
- Server must be able to run node.js applications
- Server must have MySQL installed

3. Specific Requirements

3.1 External Interfaces

3.1.1 Server interfaces

- Connection to RDBMS

3.1.2 Client interfaces

- Persistent connection to server
- Send requests to GPU
- Query the browsers DOM
- Request resources from various sources online
- Get cursor position and clicks

3.2 Functional Requirements

3.2.1

Use Case Name:	Make connection to server
Actor:	Client, Server
Priority:	Essential
Trigger:	User opens webpage
Precondition:	User is connected to the internet and on the game page
Basic Path:	1. User's client forms TCP connection with server 2. User opens game page
Alternate Path:	1. Failed to form TCP connection 1.1. Inform user that client was unable to form a connection 1.2. Try again after a few seconds 1.3. Inform user that client was still unable to connect, and that the problem lies either in their internet or the server being down.
Postcondition:	User client is connected and able to quickly send data to server
Exception Path:	

3.2.2

Use Case Name:	Login
Actor:	Client, Server
Priority:	Optional
Trigger:	User clicks login button on top right of main page
Precondition:	(3.2.1) Make connection to server
Basic Path:	1. User enters their account information 2. User clicks login
Alternate Path:	1. Failed to login 1.1. Inform user that the username and password combination was not found 1.2. Allow the user to try logging in again
Postcondition:	User's account information should show on the player information section on the webpage
Exception Path:	

3.2.3

Use Case Name:	Start game
Actor:	Client
Priority:	Essential
Trigger:	User clicks play button in button area
Precondition:	(3.2.1) Make connection to server, (3.2.15) Read mouse input
Basic Path:	1. User clicks on play button
Alternate Path:	
Postcondition:	Game starts
Exception Path:	

3.2.4

Use Case Name:	Send moves/bets
Actor:	Client, Server
Priority:	Essential
Trigger:	User selects any of the moves from the button area
Precondition:	(3.2.3) Start game
Basic Path:	1. User clicks any of the move options in button area
Alternate Path:	
Postcondition:	Client sends that move and bet combination over to the server through existing connection
Exception Path:	

3.2.5

Use Case Name:	Send chat
Actor:	Client, Server
Priority:	Essential
Trigger:	User clicks the send message button inside the chat
Precondition:	(3.2.1) Make connection
Basic Path:	1. User types text into chat field 2. User hits enter/clicks send
Alternate Path:	
Postcondition:	Client sends text over to server
Exception Path:	

3.2.6

Use Case Name:	Send gamestate
Actor:	Client, Server
Priority:	Essential
Trigger:	Server receives moves from user
Precondition:	(3.2.4) Send moves/bets
Basic Path:	1. Receives move from user 2. Verifies that move is valid 3. Sends updated gamestate to all players of that game
Alternate Path:	
Postcondition:	All connected players have screens updated
Exception Path:	

3.2.7

Use Case Name:	Update chat messages
Actor:	Client, Server
Priority:	Essential
Trigger:	Server receives chat text from client
Precondition:	(3.2.5) Send chat
Basic Path:	1. Receive chat from client 2. Send messages to all clients connected to game
Alternate Path:	
Postcondition:	All connected players see chat from all other connected players
Exception Path:	

3.2.8

Use Case Name:	Join game
Actor:	Client, Server
Priority:	Essential
Trigger:	Multiple players choose to play a game
Precondition:	(3.2.3) Start game
Basic Path:	1. Multiple users click play 2. All of them get assigned to a game number 3. In game actions are matched up with other players in same game number
Alternate Path:	1. Failed to find other players 1.1. Prompt User that no players found 1.2. Prompt User if solo game is acceptable.
Postcondition:	All players in a game receive the same game information from the server as each other
Exception Path:	

3.2.9

Use Case Name:	Handle dropped players
Actor:	Client, Server
Priority:	Low
Trigger:	A player loses their connection to the game server
Precondition:	(3.2.8) Join game
Basic Path:	1. A player's connection to the server is lost 2. Player's bet is lost, turns get skipped, cards are all sent to discard pile 3. Game continues for remaining players normally
Alternate Path:	
Postcondition:	Game runs for remaining players, dropped player loses what they bet
Exception Path:	

3.2.10

Use Case Name:	Modify game settings
Actor:	Client
Priority:	Essential
Trigger:	User clicks settings in the button area
Precondition:	(3.2.1) make connection, (3.2.15) Read mouse input
Basic Path:	1. User clicks settings and enters settings menu 2. User changes available settings 3. User clicks save changes
Alternate Path:	
Postcondition:	New settings take effect for that user
Exception Path:	

3.2.11

Use Case Name:	Determine bet
Actor:	Client
Priority:	Essential
Trigger:	Click the bet more or bet less buttons
Precondition:	(3.2.1) make connection, (3.2.15) Read mouse input
Basic Path:	1. User clicks one of the change bet buttons 2. Client changes the bet value accordingly
Alternate Path:	
Postcondition:	Bet value is changed to what user wants, 0 is a valid bet
Exception Path:	

3.2.12

Use Case Name:	Display rules (not yet implemented)
Actor:	Client
Priority:	Essential
Trigger:	User clicks on rules button in button area
Precondition:	(3.2.1) make connection, (3.2.15) Read mouse input
Basic Path:	1. User clicks rules button 2. An overlay pops up displaying the rules
Alternate Path:	
Postcondition:	The rules are displayed
Exception Path:	

3.2.13

Use Case Name:	Display game
Actor:	Client
Priority:	Essential
Trigger:	Connection is made
Precondition:	(3.2.1) make connection
Basic Path:	1. Get starting game page from server 2. Load up page into webGL canvas
Alternate Path:	
Postcondition:	Screen now shows
Exception Path:	

3.2.14

Use Case Name:	Display win/loss and payout (not yet implemented)
Actor:	Client
Priority:	Essential
Trigger:	Receive ending gamestate
Precondition:	(3.2.6) send gamestate
Basic Path:	1. Receive ending gamestate 2. Displays victory message to user 3. Displays amount gained to user
Alternate Path:	1. Receive ending gamestate 2. Displays losing message to user 3. Displays amount lost to user
Postcondition:	User sees whether they won or lost and how much they won/lost
Exception Path:	

3.2.15

Use Case Name:	Read mouse input
Actor:	Client
Priority:	Essential
Trigger:	User clicks on any button in the button area
Precondition:	(3.2.1) make connection
Basic Path:	1. User clicks on a button 2. That button's command runs
Alternate Path:	
Postcondition:	Client runs code specific to button clicked
Exception Path:	

3.2.16

Use Case Name:	User Login
Actor:	Server
Priority:	Optional
Trigger:	User logs in
Precondition:	(3.2.1) make connection
Basic Path:	1. User sends login information 2. Queries database for username and password to compare 3. Compares given username and password with stored
Alternate Path:	
Postcondition:	User is logged in
Exception Path:	User is told username and password are incorrect

3.2.17

Use Case Name:	Validate bets/moves
Actor:	Server, Client
Priority:	Essential
Trigger:	Client sends bets/moves
Precondition:	(3.2.4) Send bets/moves, (3.2.16) query database
Basic Path:	1. Receive bets/moves 2. Query database for their cash and potential moves 3. Determine to be valid 4. Update gamestate
Alternate Path:	1. Determine to be invalid 2. Request player to redo move
Postcondition:	Moves are determined valid and applied.
Exception Path:	

3.2.18

Use Case Name:	Shuffle deck
Actor:	Server
Priority:	Essential
Trigger:	Game started
Precondition:	(3.2.3) Start game
Basic Path:	1. Game is started 2. Determines random seed 3. Shuffles deck with created seed
Alternate Path:	
Postcondition:	Deck is randomly shuffled for every game
Exception Path:	

3.2.19

Use Case Name:	Display opponent visible cards
Actor:	Client, Server
Priority:	Essential
Trigger:	Playing a multiplayer game
Precondition:	(3.2.8) connect multiple players
Basic Path:	1. Server delivers game state 2. Client displays opponent cards in player bar
Alternate Path:	
Postcondition:	User sees all shown cards of opponents
Exception Path:	

3.2.20

Use Case Name:	Play against AI
Actor:	Server
Priority:	Low
Trigger:	User plays a game alone
Precondition:	(3.2.3) Start game
Basic Path:	1. User starts game 2. Server runs a decision tree based AI to make their moves against the player
Alternate Path:	
Postcondition:	Player is playing against ai.
Exception Path:	

3.2.21

Use Case Name:	Load buttons dynamically
Actor:	Client
Priority:	Essential
Trigger:	User selects and joins a game
Precondition:	(3.2.8) Join game
Basic Path:	1. User joins game 2. Client loads buttons as described in json sent by server
Alternate Path:	
Postcondition:	Player can now click buttons to make their moves
Exception Path:	

3.2.22

Use Case Name:	Select game
Actor:	Client
Priority:	Essential
Trigger:	User selects a running game
Precondition:	(3.2.1) Connect to server
Basic Path:	1. User connects to server asking for game 2. Server responds with open games 3. User selects one to join and notifies server
Alternate Path:	
Postcondition:	Player joined their selected game
Exception Path:	

3.3 Performance Requirements

- Client should be able to render the scene at at least 30 FPS
- Matchmaking should take less than 3 minutes to complete
- Game loading on subsequent visits should load within 1 minute (praise the cache)
- The time between a player making a move and other players updating the game state should be under 5 seconds
- Visual response to user input should be within 100 milliseconds

3.4 Logical Database Requirements

- Client
    - Contains client username, hash, and total credits

3.5 Design Constraints

- Screen ratio and resolution - ideally our game should fit entirely within the user's screen
- Device whether phone (optional) or other (tablet, laptop, or desktop)
- Client-server architecture
- Web-based
- Security of transactions and game play

3.6 Software System Attributes

- Reliability
    - The platform and example games shouldn't crash the system.
    - Future developer made games shouldn't crash the system.
- Accessibility
    - The system should have support for colorblind people.
    - The system should have support for blind people (later).
    - Deaf people can play without issue.
- Availability
    - The application should be available for the user at all times.
    - Multiplayer game availability will depend on number of users on at any given time and how many are accepting new players.
- Security
    - Users shouldn't be able to easily cheat.
    - Protects user information.
- Maintainability
    - The platform should be easy to extend.
    - The platform should be modular.
- Portability
    - The user can access the game from any computer with an internet connection.