

Design Document

WebGL Card Game Platform

Version 1.6

3/15/19

Team WebGL Game

https://docs.google.com/document/d/1JLMnEJU9RaDpTXMWtOYVJ_XVxs6VLg2dP8r2Bk2ku-8/edit

Computer Science / 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	11/7/18
1.1	Jeffrey Talada, Sean Hardin	Finalized for Milestone 1	12/4/18
1.2	Jeffrey Talada, Anthony Klobas	Updated class and dependency diagrams	1/24/19
1.3	Anthony Klobas	Updated class diagrams, dependency diagram	2/8/2019
1.4	Jeffrey Talada	Updated class diagrams	2/20/19
1.5	Jeffrey Talada	Updated traceability matrix	3/8/19
1.6	Anthony Klobas, Sean Hardin, Jeffrey Talada	Updated database and dependency descriptions, DFD level 2 diagram; Added DFD level 1, Component, and Database schema diagrams	3/15/19

Table of Contents

Revisions	2
Table of Contents	3
1. Introduction	4
1.1 Purpose	5
1.2 Scope	5
1.3 Definitions, Acronyms and Abbreviations	5
1.4 References	6
2. System Overview	7
3. System Components	8
3.1 Decomposition Description	9
3.1.1 Server	9
3.1.2 Client	9
3.2 Dependency Description	11
3.3 Interface Description	12
3.3.1 Module Interfaces	12
3.3.2 User Interfaces (GUI)	12
Login Screen	12
Rules Screen	13
Settings Screen	14
Game Play Layout	15
Game Play Area	16
Chat Area	16
Player Information Area	17
Settings, Rules, Game Play Button Area	17
4. Detailed Design	18
4.1 Class Diagrams	18
4.1.1 Full Diagram	18
4.1.2 Server Diagram	19
4.1.3 View Diagram	20
4.1.4 Controller Diagram	21
4.1.5 Model Diagram	22
4.2 Component Diagram	22
4.3 Module Detailed Design	23

4.3.1 Data-flow Diagram Level 1	23
3.4.2 Data-flow Diagram Level 2	24
4.4 Database Schema	25
4.5 Requirements Traceability Matrix (RTM)	25

1. Introduction

1.1 Purpose

The purpose of this document is to describe how to architect an online card game, including how to separate the particular ruleset from the GUI and the server. It is intended to help the implementers and Sara Farag.

1.2 Scope

The current scope of our software is to build an online card gaming platform that utilizes WebGL graphics. It will support a modular design allowing for easily updating individual functionalities, and swapping out certain visuals or functions as a user desires. The modules we are currently working on include basic visuals, loading assets from file, loading the game rules from file, and functions to pass only the needed information in order for game logic to be completely separate from display logic.

The benefits of our platform lie in the modularity. When we add new functionalities in the future, we will only need to modify one of the existing modules, rather than keep track of all the different parts of a full game that need to be changed.

1.3 Definitions, Acronyms and Abbreviations

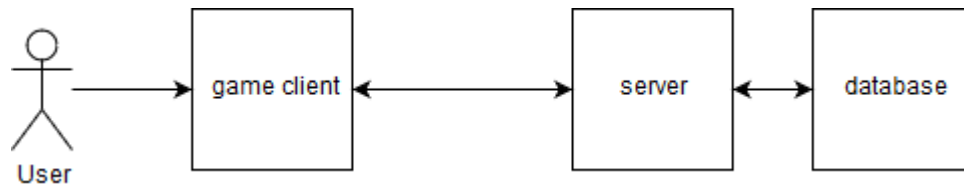
- AI - artificial intelligence
- Client - The application running inside of a user's web browser that communicates with our server to run the game
- DOM - Document object model
- FPS - Frames per second
- Fragment - Small section of a polygon to be rendered (think pixel)
- GLSL - OpenGL Shading Language
- GPU - Graphics processing unit
- IDE - Integrated Development Environment
- JS - Java Script
- MVC - Model View Controller, a common way of decoupling the code that handles interaction, logic, and display
- 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

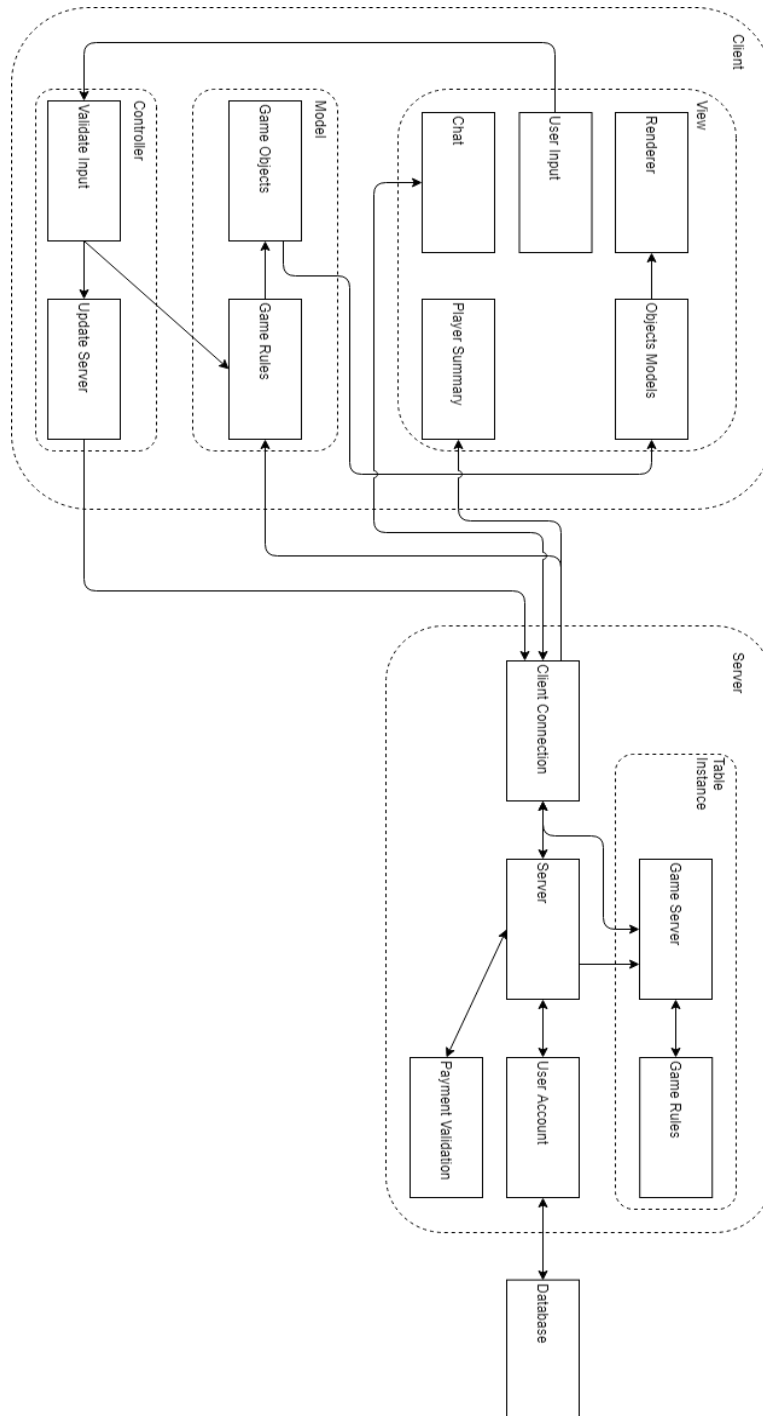
- Client-Server: wikipedia.org, 'Client-server model', 2018. [Online]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model. [Accessed: 3- Dec- 2018].
- Microservice: microsoft.com, 'Microservices architecture style', 2018. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>. [Accessed: 3- Dec- 2018].
- MVC: mozilla.org, 'MVC architecture', 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture. [Accessed: 3- Dec- 2018].
- N-Tier: microsoft.com, 'N-tier architecture style', 2018. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier>. [Accessed: 3- Dec- 2018].
- Node.js: nodejs.org, 'About Node.js'. [Online]. Available: <https://nodejs.org/en/about/>. [Accessed: 3- Dec- 2018].
- Three.js: threejs.org, 'Featured Projects'. [Online]. Available: <https://threejs.org/>. [Accessed: 3- Dec- 2018].

2. System Overview



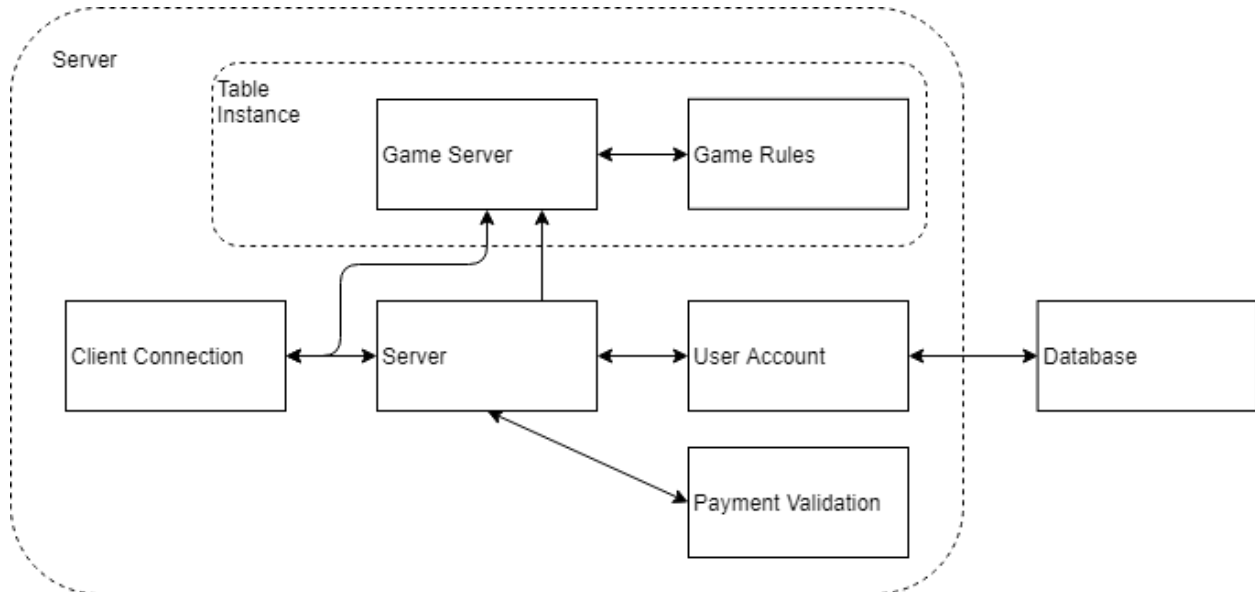
3. System Components

- Overall architecture is client-server
- Client is MVC
- Server is a collection of Microservices
 - i.e. game, chat, and payment are independent services



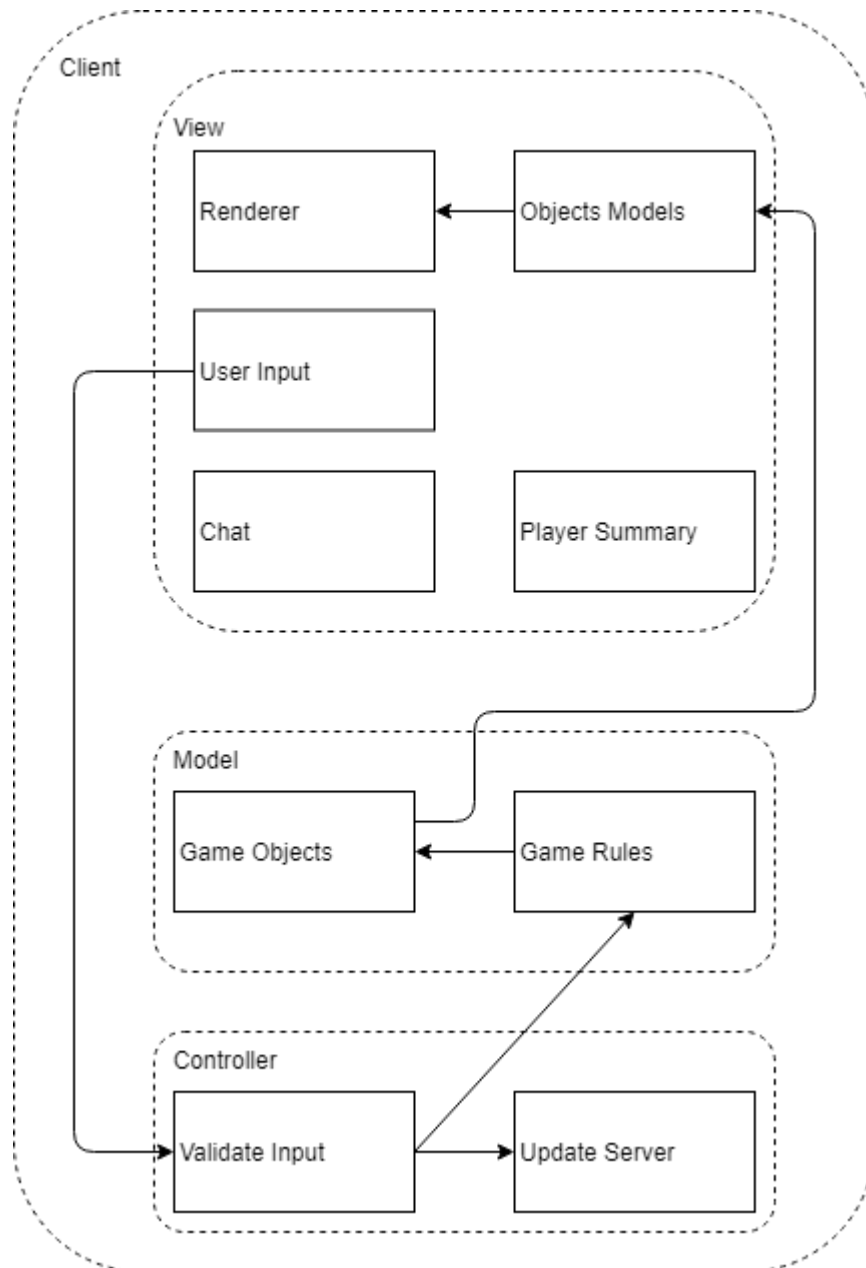
3.1 Decomposition Description

3.1.1 Server



- A client will connect to the server through the Client Connection
 - The Server then spawns a Game Server and connects it to the client through the Client Connection
- A client may maintain a User Account while connected to the Server
 - User Account information is stored in a Database
 - The Game Server spawns a copy of the desired Game Rules
 - Game Server broadcasts game updates to each player's client through Client Connection
- Game Rules validates a client's actions

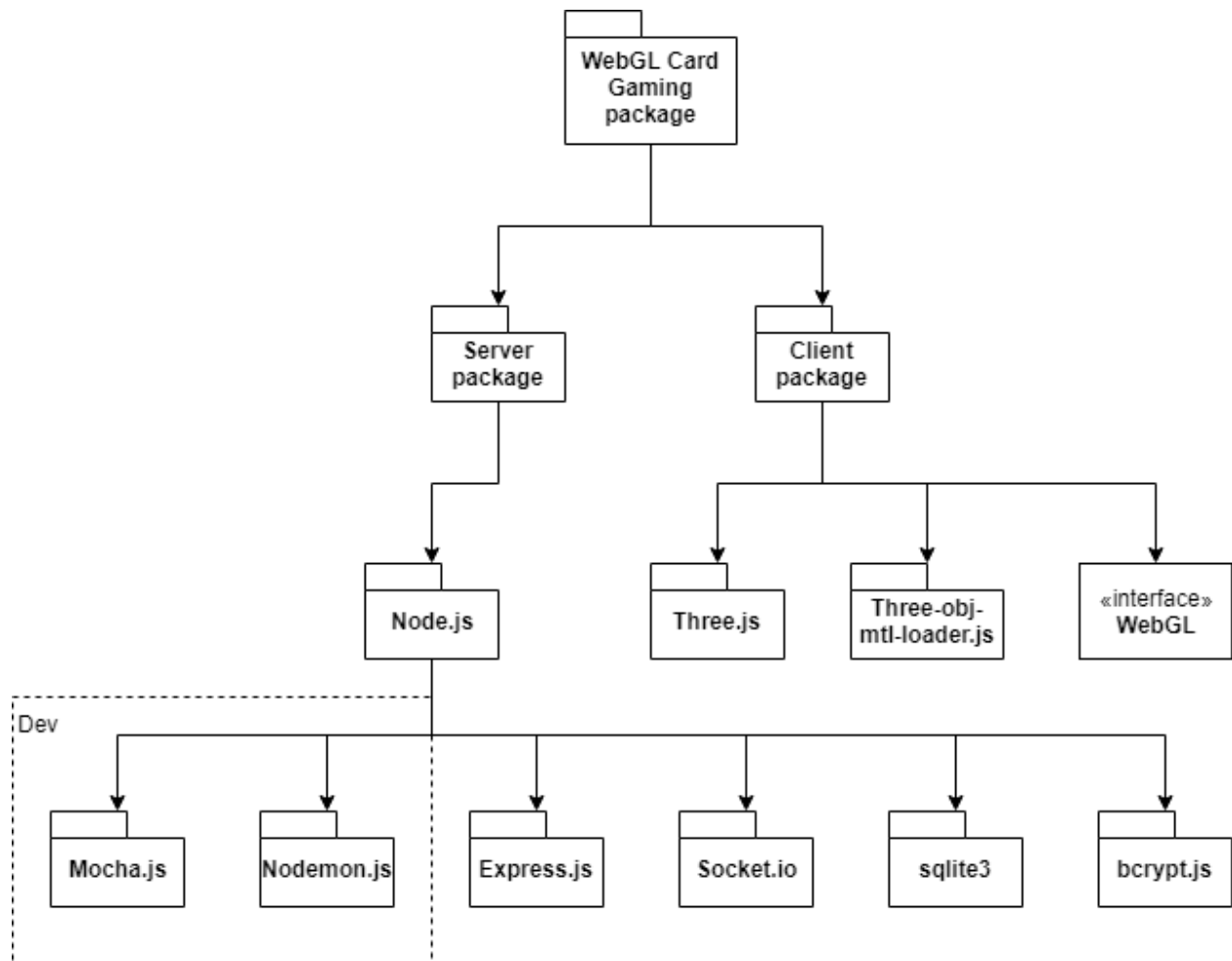
3.1.2 Client



- When the client is connected to a Game Server the Games Rules are updated by the Server along with turn information
- The Game Rules determine which Game Objects can be interacted with and in what ways and sends the data to the Object Models
- Object Models tell the Renderer where they are and whether they should be highlighted based on the Game Rules
- User Input includes buttons and mouse clicking and dragging
- User Input sends button presses and mouse clicks to Validate Input

- Validate Input forwards the button presses and mouse clicks to Update server for server-side validation and to update other players
- Validate Input sends the moves to Game Rules which updates the game state while waiting for server validation
- Chat sends information to the server which broadcasts messages to other players
- The Server sends game updates to Player Summary

3.2 Dependency Description



The whole system breaks down into the Server package and Client package. In order for the Client side to display the game in 3D, Three.js, Three-obj-mtl-loader.js, and a computer that can at least emulate a graphics card. The Server package runs in the Node.js environment and the code requires Express.js, Socket.io, sqlite3, and bcrypt.js in order to function with the client. Mocha.js allows for unit testing and Nodemon.js allows for quick iterative development without having to manually restart the server after every change.

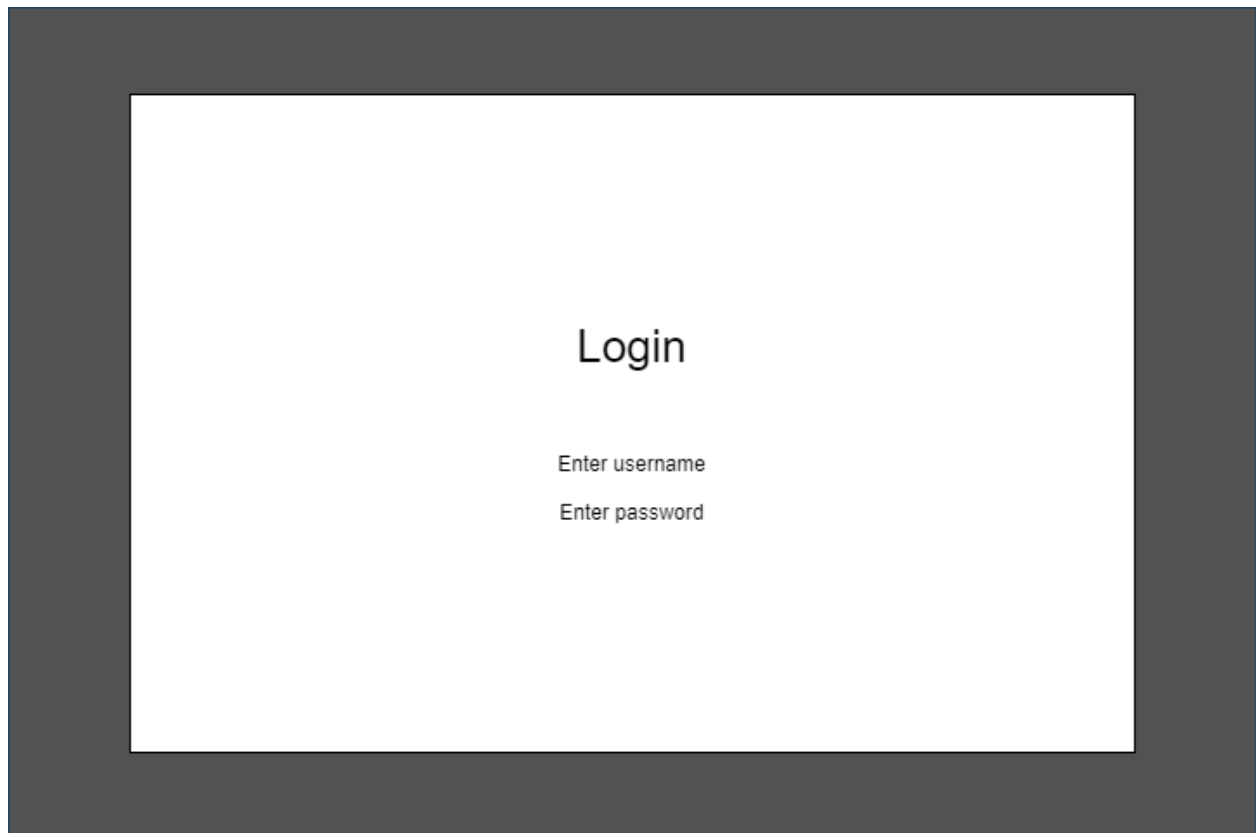
3.3 Interface Description

3.3.1 Module Interfaces

- External Interfaces
 - Payment Processing (not yet implemented)
- Internal Interfaces
 - Client - Server
 - The Client sends:
 - Player moves, bets, and chat (partially implemented)
 - The Server broadcasts:
 - Game state updates including player moves and bets
 - Chat
 - The server communicates with a database for user account information

3.3.2 User Interfaces (GUI)

Login Screen



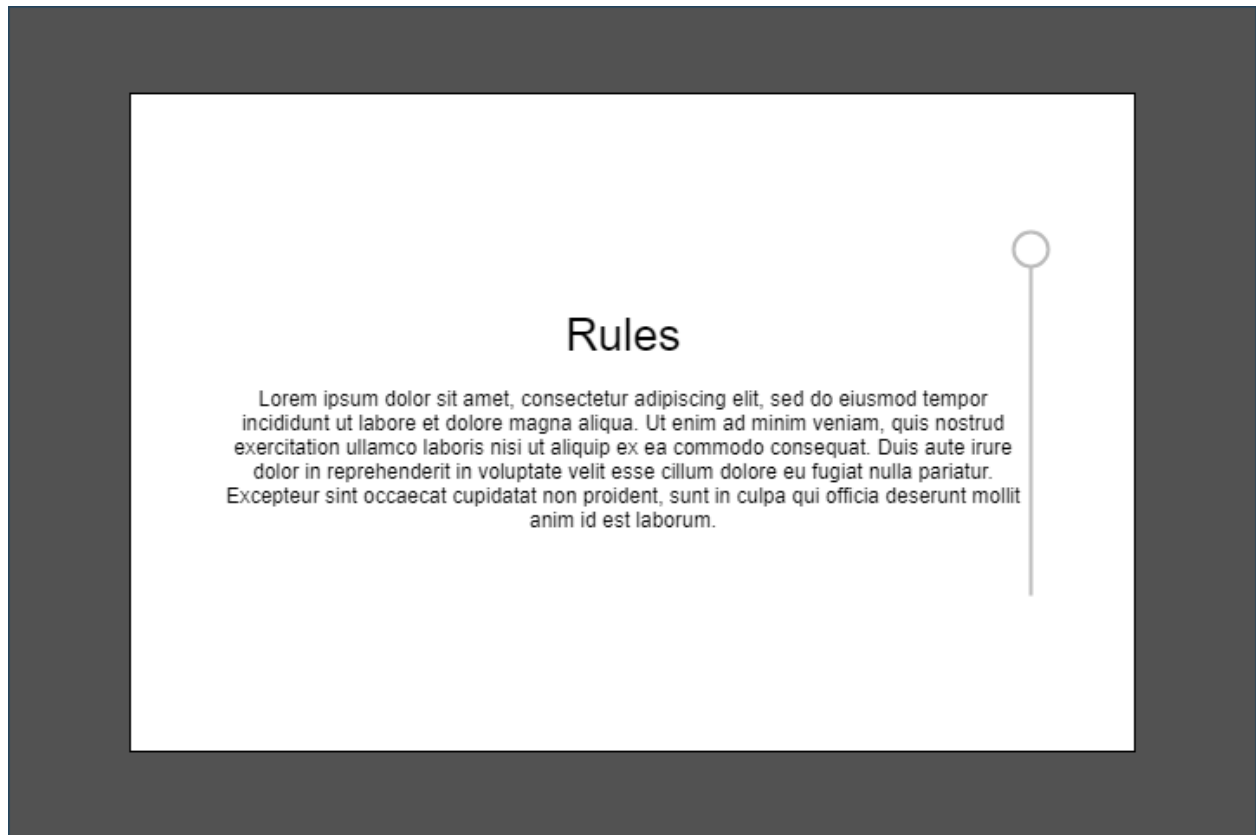
Login

Enter username

Enter password

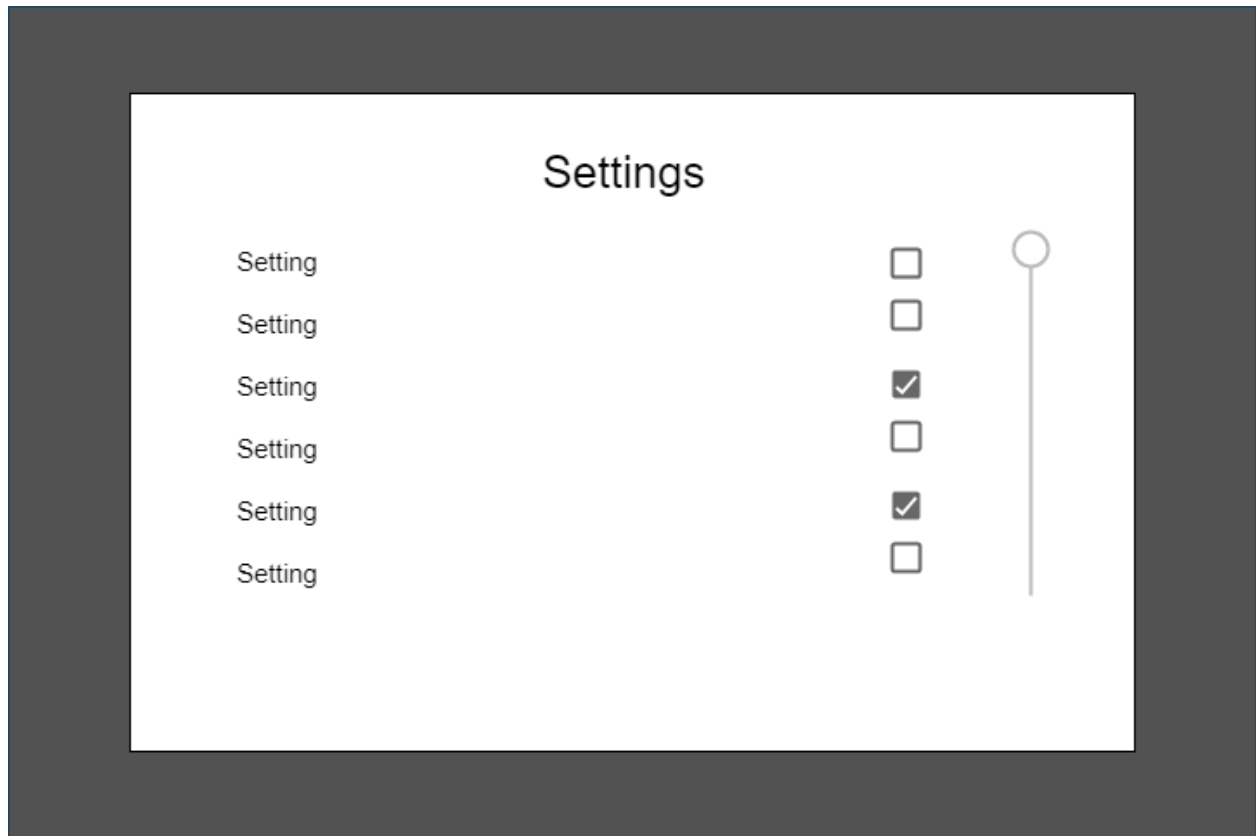
Users will be greeted with a login screen that asks for a username and password.

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

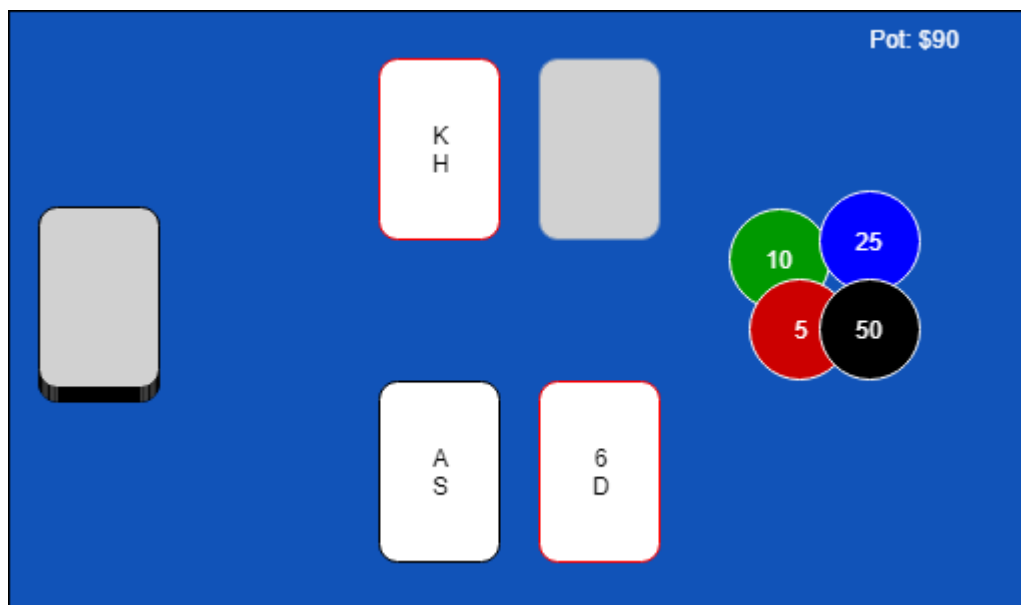


Should a user be interested in non-default game play, they can alter the game rules through the settings screen by checking and unchecking boxes.

Game Play Layout

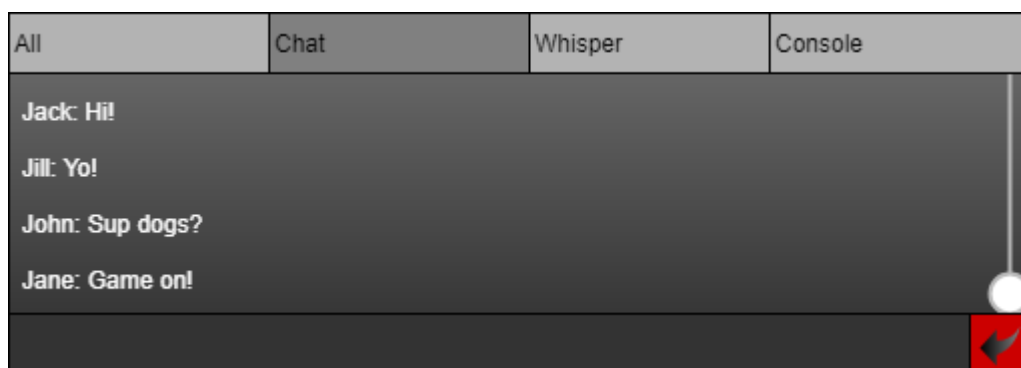
Game Play Area	Player and other player information display (Banks, bets, current cards)
Chat	Settings, Rules, Game play buttons

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

Bet: \$27	Bank: \$250
 Jack	   
Bet: \$32	Bank: \$150
 Jane	   
Folded	Bank: \$187
 John	   

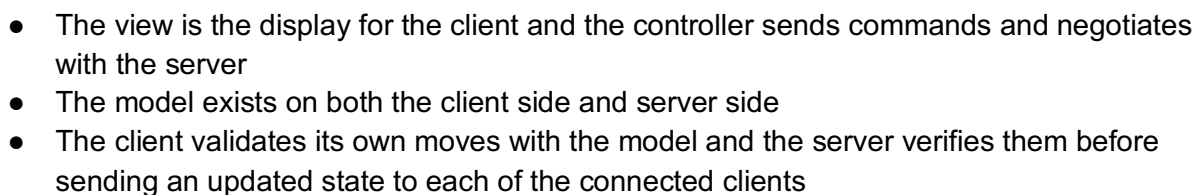
The player information area contains a summary of all the player's states. It shows their current bets, banks, and hands.

Settings, Rules, Game Play Button Area

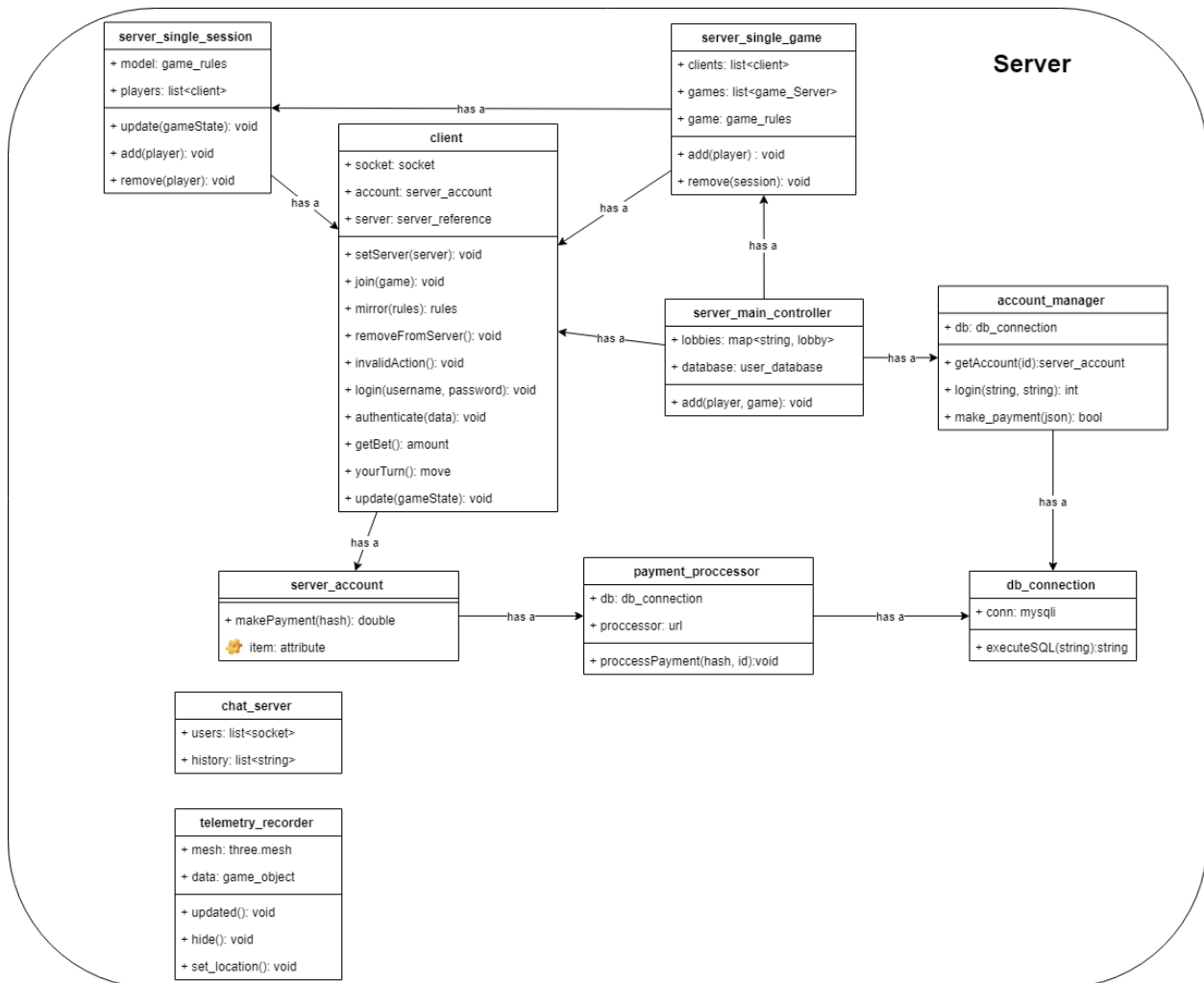
ALL IN	
Hit	Fold
Split	Draw Card
Bet More	Bet Less
Rules	Settings

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.

4.1.1 Full Diagram

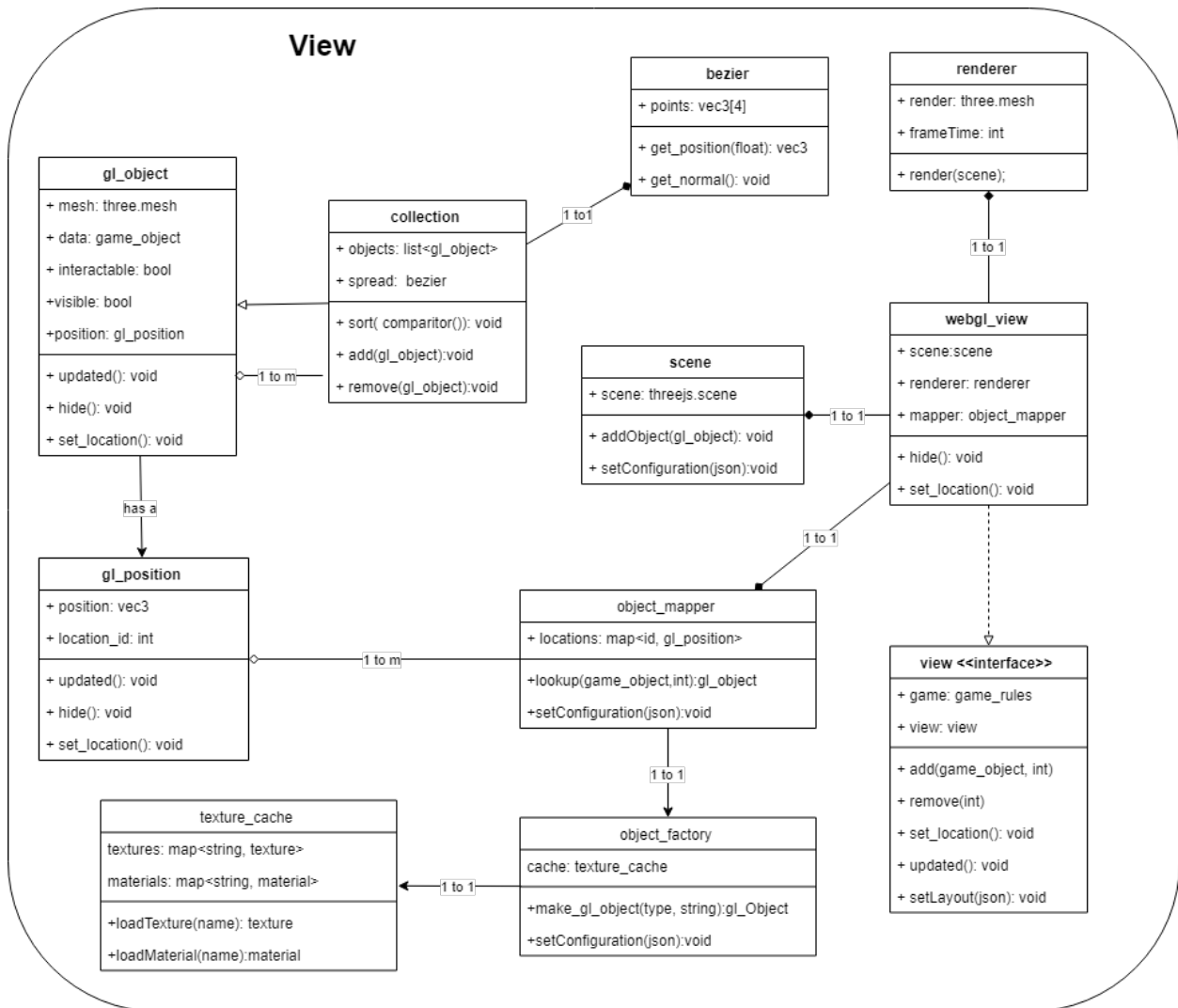


4.1.2 Server Diagram



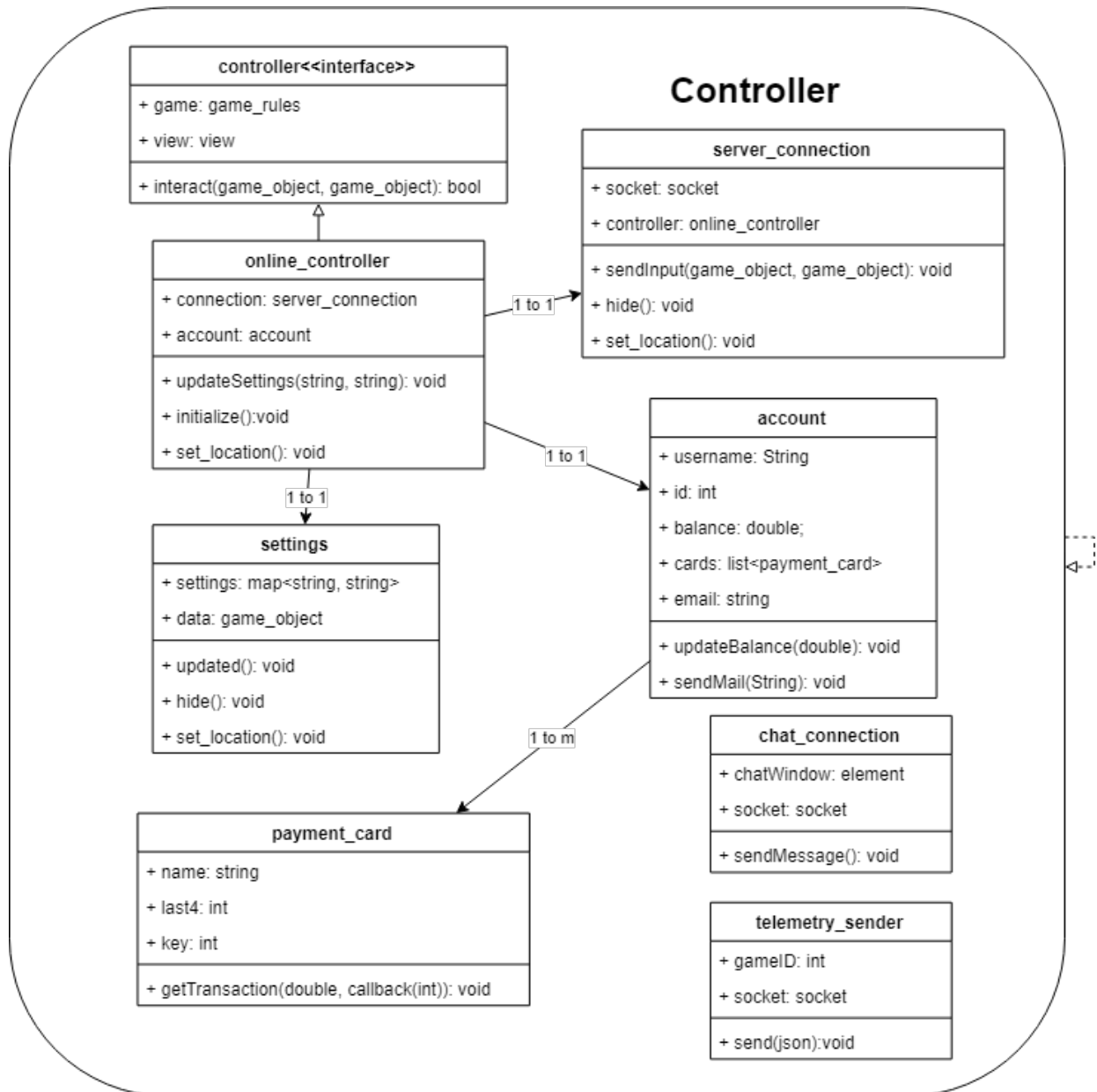
- Server main controller has a server single game for each type of game playable
 - When a client requests to join a game main controller passes the client to server single game
 - Server single controller has every currently running instance of that gametype and inserts the client into one that's not full, server single session
- Server single session implements game rules in Model and is a component of server single game
- Client talks to server connection in Controller
- Server account talks to account in Controller
- Chat server talks to chat connection in Controller
- Telemetry Recorder receives data from telemetry sender in Controller (not yet implemented)
- Payment processor (not yet implemented)

4.1.3 View Diagram



- Collection is both a `gl_object` and a collection of them (theoretically it can contain instances of collection too)
- `Object_mapper` keeps track of what `gl_objects` are in the scene and uses its lookup function to find the `gl_position` to add new objects at. The positions are all defined when `setConfiguration` is called.
- Game rules in `Model` tells view what to display
- View has a connect to server connection in `Controller`

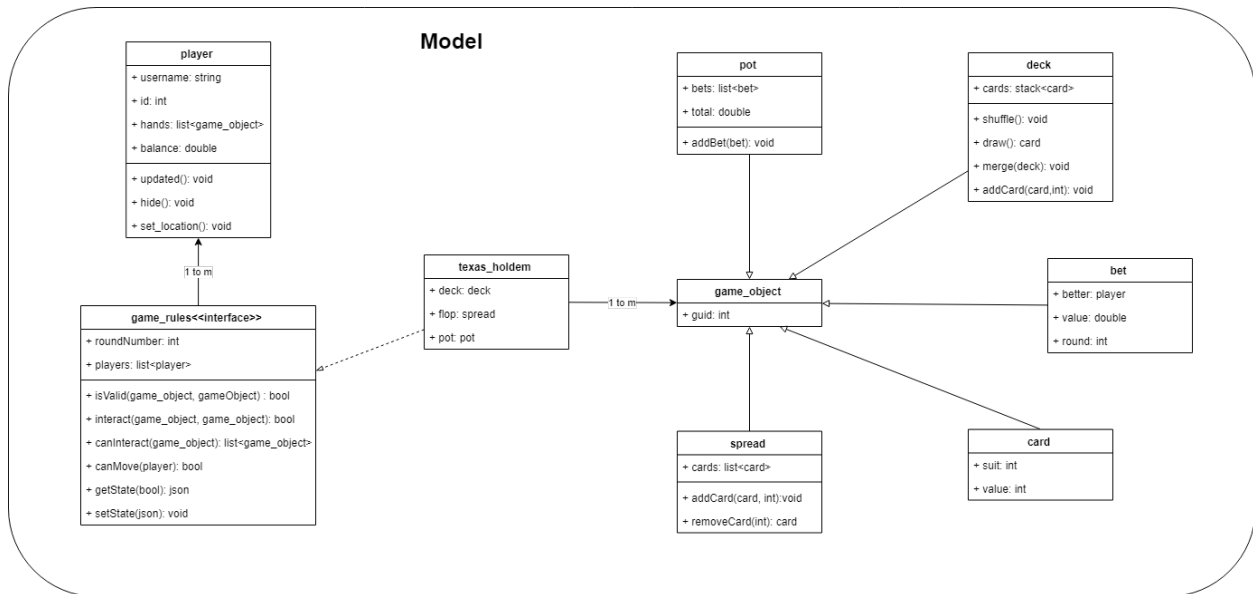
4.1.4 Controller Diagram



- Chat connection
- Telemetry sender (not yet implemented)
- Payment card (not yet implemented)
- View in View has a connection to server connection
- Client in Server talks to server connection
- Server account in Server talks to account
- Chat server in Server talks to chat connection

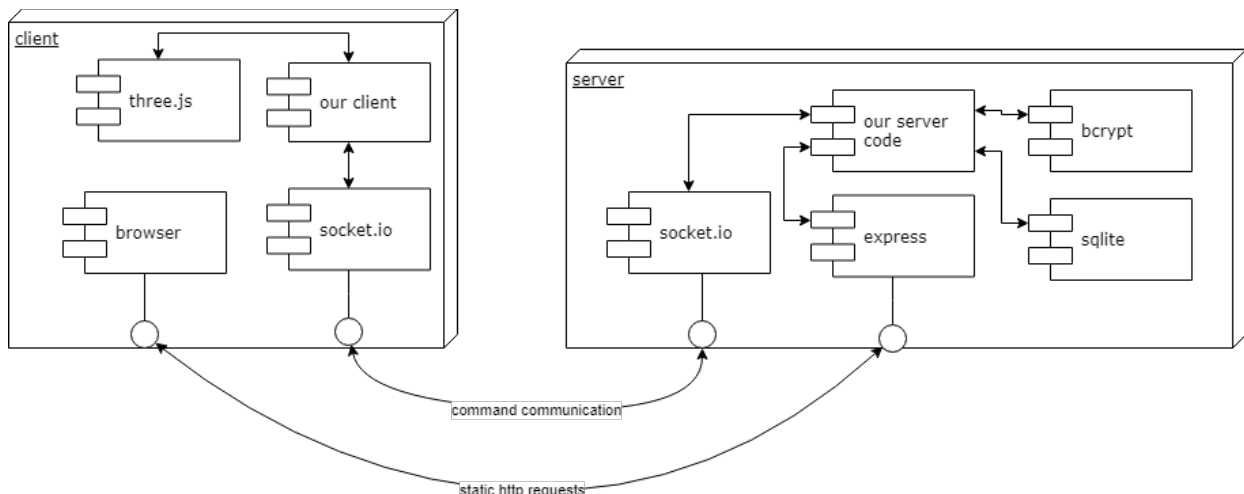
- Telemetry Recorder in Server receives data from telemetry sender (not yet implemented)

4.1.5 Model Diagram



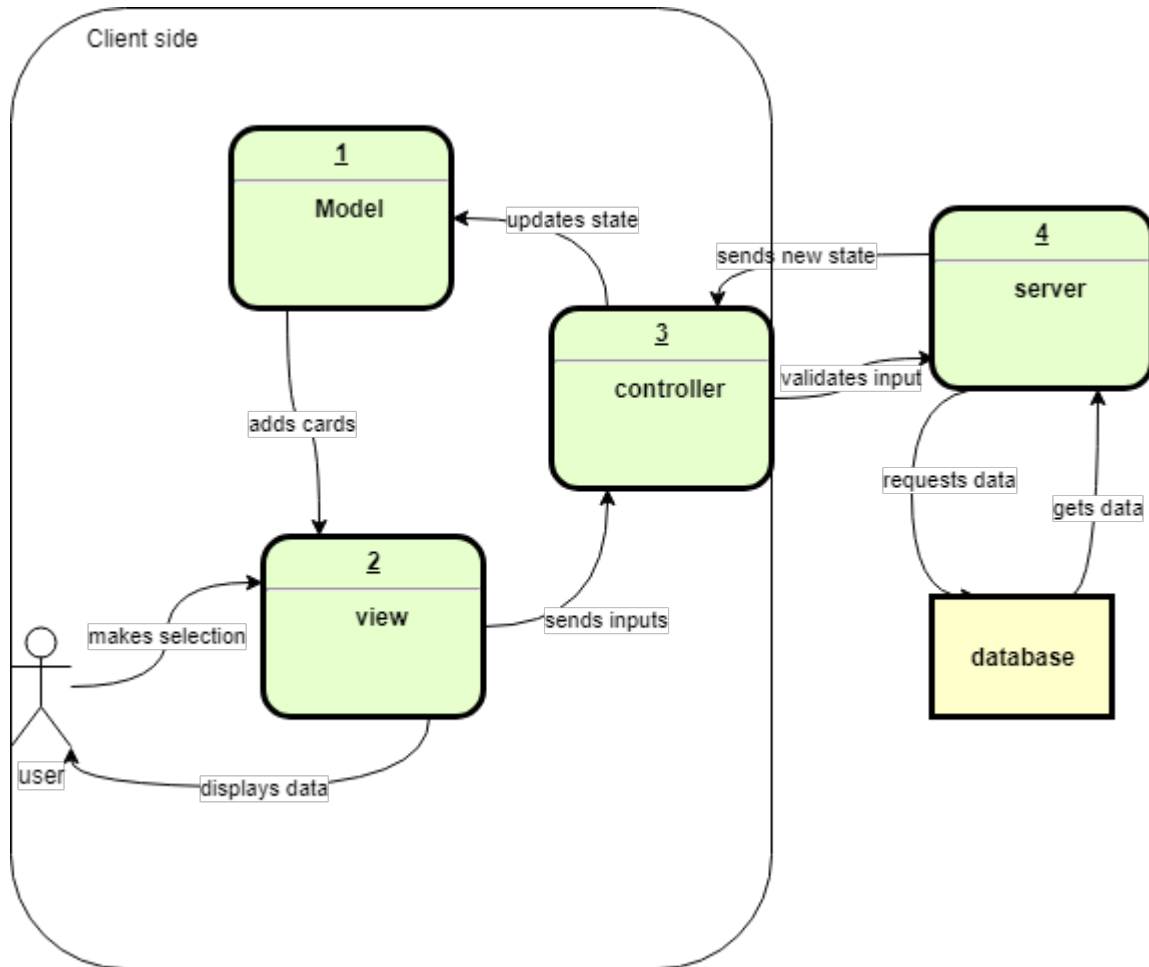
- The model is unique to the game type a player chooses, and can be swapped for any other.
 - This particular one has a theoretical version of texas hold 'em as an example
- Server single session in Server implements game rules and is a component of server single game in Server
- Game rules tells view in View what to display
- Game rules receives instructions from server connection in Controller

4.2 Component Diagram



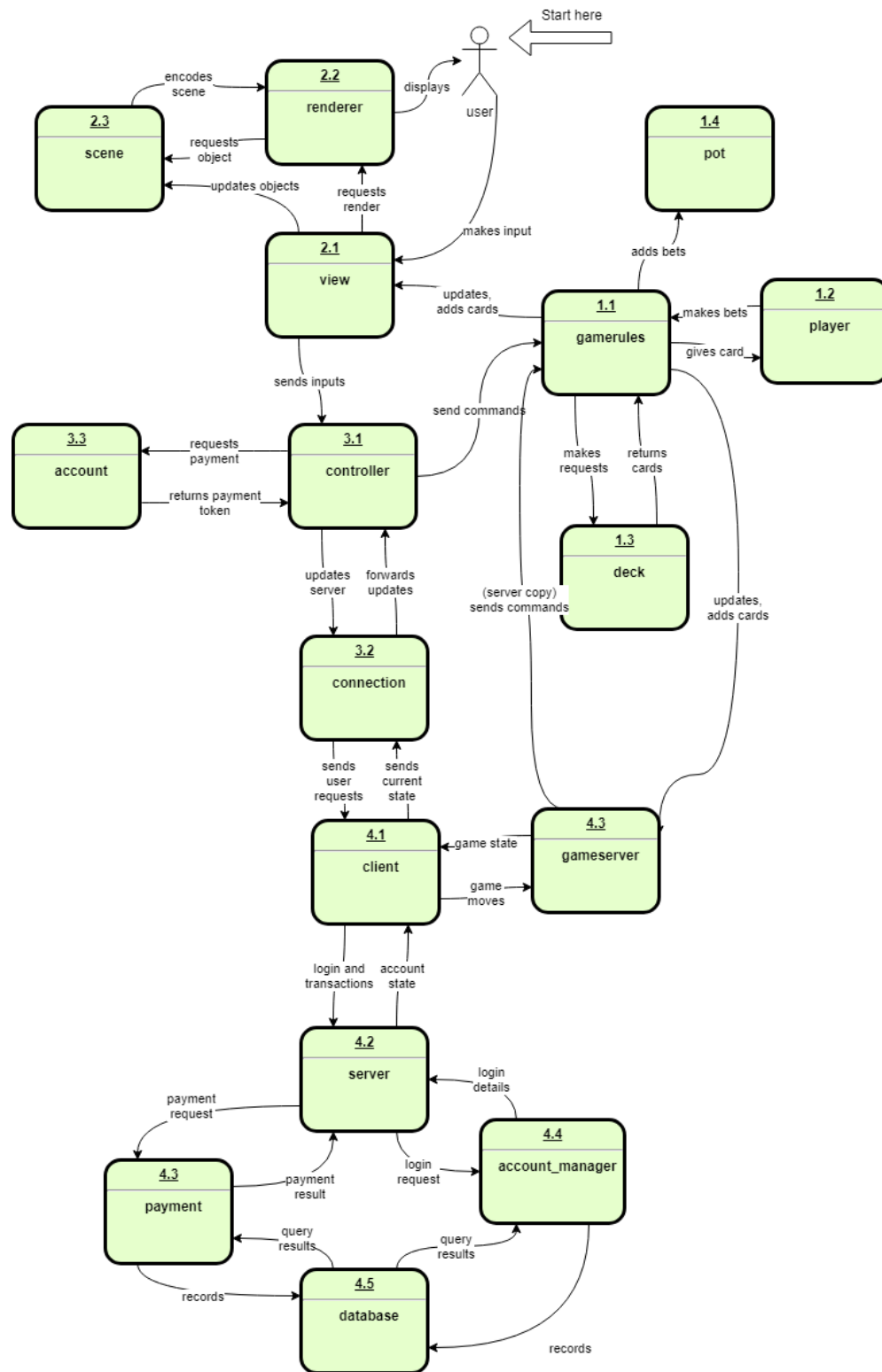
4.3 Module Detailed Design

4.3.1 Data-flow Diagram Level 1



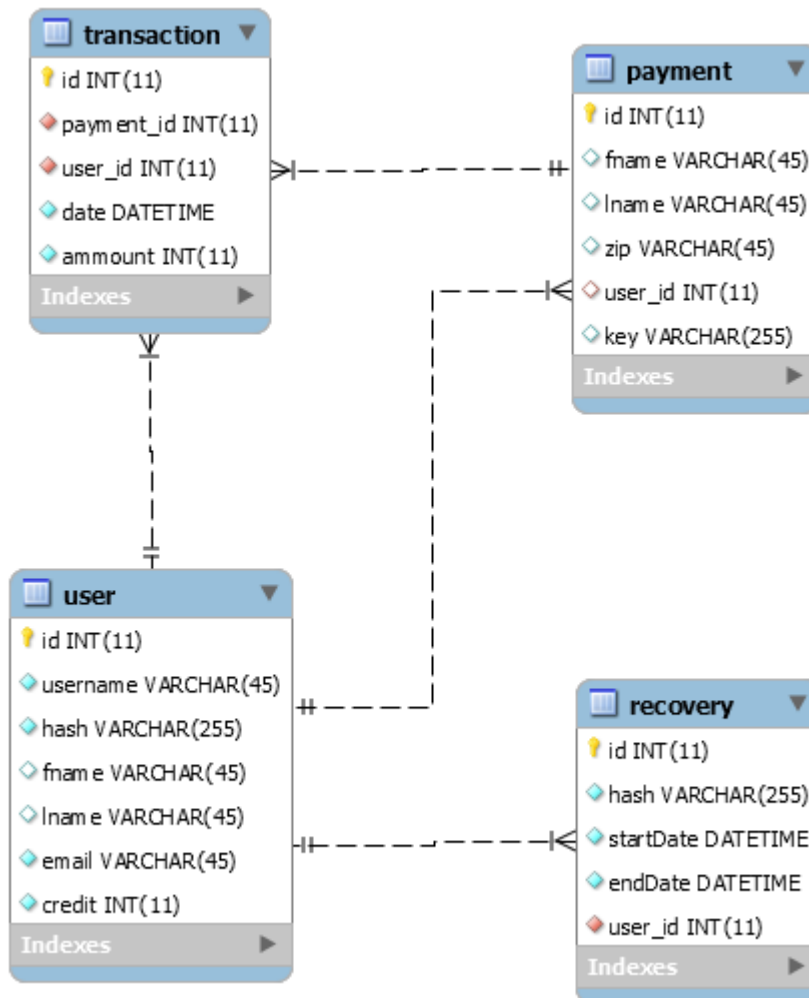
Overall flow inputs flow from the user through the view to the controller. The controller then directs the input at the model or server. Then the controller updates the view which sends information to the user.

3.4.2 Data-flow Diagram Level 2



Expanded interactions from level 1 Data Flow Diagram
Start at user module 2.1

4.4 Database Schema



- User table contains user data
- Payment table contains payment info such as credit cards*
- Transactions are specific payments from a credit card for a user*
- Recovery is for account recovery*

* not implemented

4.5 Requirements Traceability Matrix (RTM)

Requirement #	Requirement	Module Design Component	Data Design Component (not fully implemented yet)
3.2.1	Make connection to server	3.2	Server

3.2.2	Login	4.4	Client
3.2.3	Start game	4.3	GameServer
3.2.4	Send moves/bets	3.1	Playerhand, playerbet
3.2.5	Send chat	3.2	n/a
3.2.6	Send gamestate	4.1	GameServer
3.2.7	Update chat messages	4.1	n/a
3.2.8	Join game	4.2	GameServer
3.2.9	Handle dropped players	4.2	Playerhand
3.2.10	Modify game settings	3.1	Client
3.2.11	Determine bet	2.1	Client
3.2.12	Display rules	2.2	n/a
3.2.13	Display game	2.2	Renderer
3.2.14	Display win/loss and payout	2.2	View
3.2.15	Read mouse input	2.1	View
3.2.16	Query database	4.5	Server
3.2.17	Validate bets/moves	4.3/1.1	GameServer
3.2.18	Shuffle deck	1.3	Deck
3.2.19	Display opponent visible cards	2.2	GameServer
3.2.20	Play against AI	4.3	GameServer
3.2.21	Load buttons dynamically	2.1	View
3.2.22	Select game	3.1	Server, Client