

# **CSC667 Spring 2017**

## **Application: CaeZar**

2017/05/20

**Team CCZZ**

Xinlu Chen (915757497)

Cheng Li(916422876)

Xuan Zhang(916409525)

Lijie Zhou(913821407)

<https://github.com/SFSU-CSC-667/term-project-spring-2017-cczz>

## **Table of Contents**

1. Project Architect-----	2
2. Database diagram-----	2
3. Technology Stack-----	3
4. Problems Solved and Challenges-----	5
5. Grading Rubric-----	6
6. Test Plan-----	7

## 1. Project Architect

Our project has four basic parts: registration/login, lobby page, game, and chat.

### Registration/Login

Users have to register and login to enter the game.

### Lobby page

The lobby page has three main parts: scoreboard, game board, and chatboard. The scoreboard is a bulletin board that displays game player's current score(s). The game board displays the current game rooms that users can enter. If the user doesn't want to enter the current room(s), he/she can choose to create a new room. The chatboard is used for users to post messages. One use case is for users to notify their friends which game to enter.

### Game page

After users enter the room, the game page will display. The game room uses canvas to show game interface. In the game page, users can use the "start game" button to start the game. Users can also use chat board to communicate with each other in the game room.

## 2. Database diagram

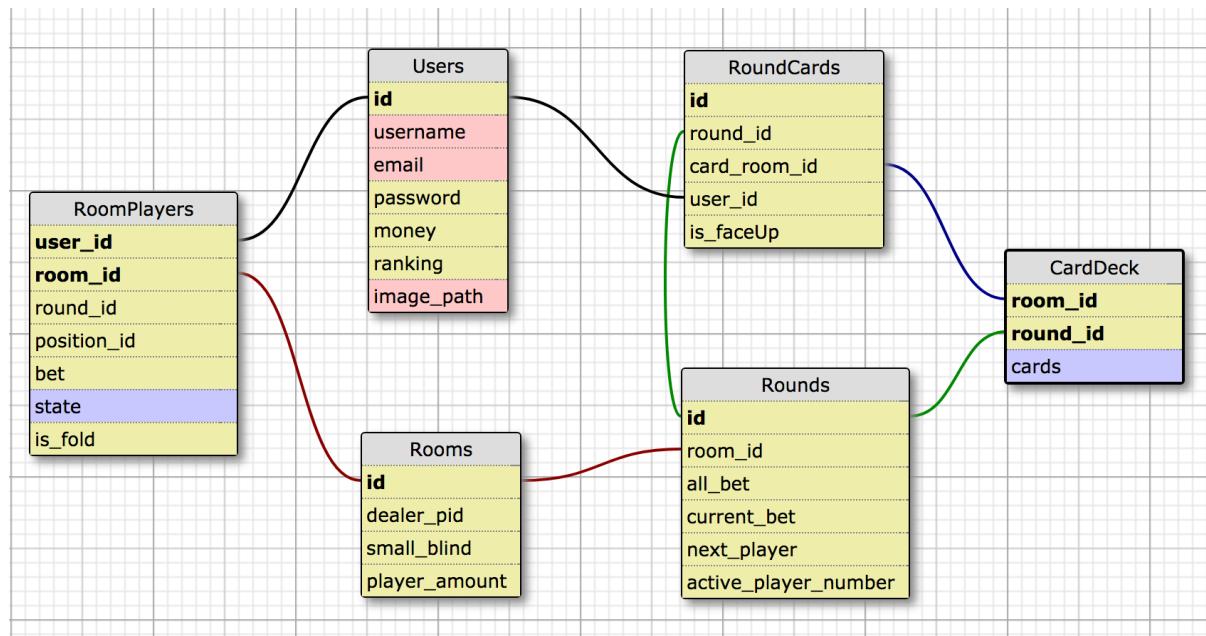


Figure 1 Database Diagram

### 3. Technology Stack

We use code snippets to show the main technologies we used in this project.

#### 3. 1 Frontend: Pug/Bootstrap , CSS, and Javascript/JQuery

We use Pug/Jade to render the views in /views folder. Here below is one example of the lobby.pug file.

```
1  extends layout
2
3  block content
4      .container.lobby-body
5          #ranking-board.col-md-4
6              h3 Score Board
7                  .score1
8
9          #game-room.col-md-4
10             h3 Game Board
11             #rooms.list-group
12
13
14         #chat-board.col-md-4
15             h3 Chat Board
16             div.display-message.well.well-sm
17
18             .input-group
19                 input.form-control(type='text',
20                     placeholder='Message...')
21                 span.input-group-btn
22                     button.btn.btn-default(type='button')
23                         Send
24
25
```

Figure 2 lobby.pug

In lobby.pug, we use Bootstrap to layout three columns as seen in *col-md-4*. The pug file is a skeleton of the user interface. We use JQuery to append the dynamic data on the pug file.

```
20     /*Fill the score board and room board*/
21     $.get("/api/rooms", function (data, status) {
22         for (var i = 0; i < data.length; i++) {
23             $('#rooms').append(listItem(data[i]));
24         }
25     });

```

Figure 3 lobby.js

### 3. 2 Backend: NodeJS, Express, Postgresql

We use socket to handle in-app communication. There are two features that use socket intensively: first, user message exchange both in lobby and game room. The second is game state update. Below, we will use the message exchange as an example to explain.

The /socket/index.js file handles all the server side socket code.

```
26 const init = function (app, server) {
27   const io = socketIo(server); // the websocket connection
28
29 //app.set('io', io); //useless
30
31   io.sockets.on('connection', function (socket) {
32     /*socket on the message from the lobby*/
33     socket.on('message', function (data) {
34       io.emit('message-display', data );
35       // console.log(data.data);
36     });
37     /*socket on the message from the room*/
38     socket.on('room-message', function(data){
39       // console.log(data.roomid);
40       socket.join(data.roomid);
41       io.sockets.in(data.roomid).emit('room-message-display', data);
42       // console.log(data.data);
43     });
44   });
45 }
```

Figure 4 /socket/index.js

The client side socket is in javascripts/lobby.js and javascripts/game.js. Here below is the code snippet of how to register the 'room-message' event from the client side's socket.

```
178 /* Room message posted */
179 $('#chat-input button').click(function () {
180   const message = $('.room-form-control').val();
181   // console.log(message);
182   // var username = $.cookie(username);
183   socket.emit('room-message', {roomid: roomid});
184   socket.emit('room-message', {data: message});
185 });

186 
```

Figure 5 /javascripts/lobby.js

The database codes are organized in /db and /model/index.js. /db stores SQL commands that will first create tables and insert records. /model/index.js stores commands that will be used to retrieve data from the database.

#### **4. Problems solved and challenges**

##### Problem 1: Heroku Database Migration

At first, we don't know how to migrate Postgresql database to Heroku if our website has already deployed on it. After reading the documentation and the hints John provided on Slack, we figured out a way to create a new postgresql database on Heroku and insert data into it.

##### Problem 2: Understanding Socket

We spent a lot time researching on how to use socket for in-app communication. One challenge is how to channel messages from the same socket in the server into different rooms on the client's side. For example, when we have to use server side messages back to the room based on its room number, we don't want players from a different room to see the messages from a different room. We found .join() is a method that specifically designed for channeling messages into different channels after extensive research.

##### Problem 3: Understanding the MVC architect

It took us a while to figure out the MVC architect and how to take data from database to populate on the page. One mistake we made early on is to hard code the pug file. Later, we figured we should first get all the data from the database and "store" it on the endpoints. For example, /api/users have all the records retrieved from the users table from the database. In order to use it from the frontend, we use JQuery .get('/api/users') to get all the records and manipulate it using JQuery. From there, we can dynamically append new records onto the pug file.

##### Problem 4: Different OS

In our group, we have 4 team members, some of us uses Mac OS and some of us uses Windows. The folder javascript in Mac is 'Javascript' and in Windows is 'javascript'. But in Heroku, it will become 2 folders—'javascript' and 'Javascript' which makes the application not work.

## 5. Grading Rubric

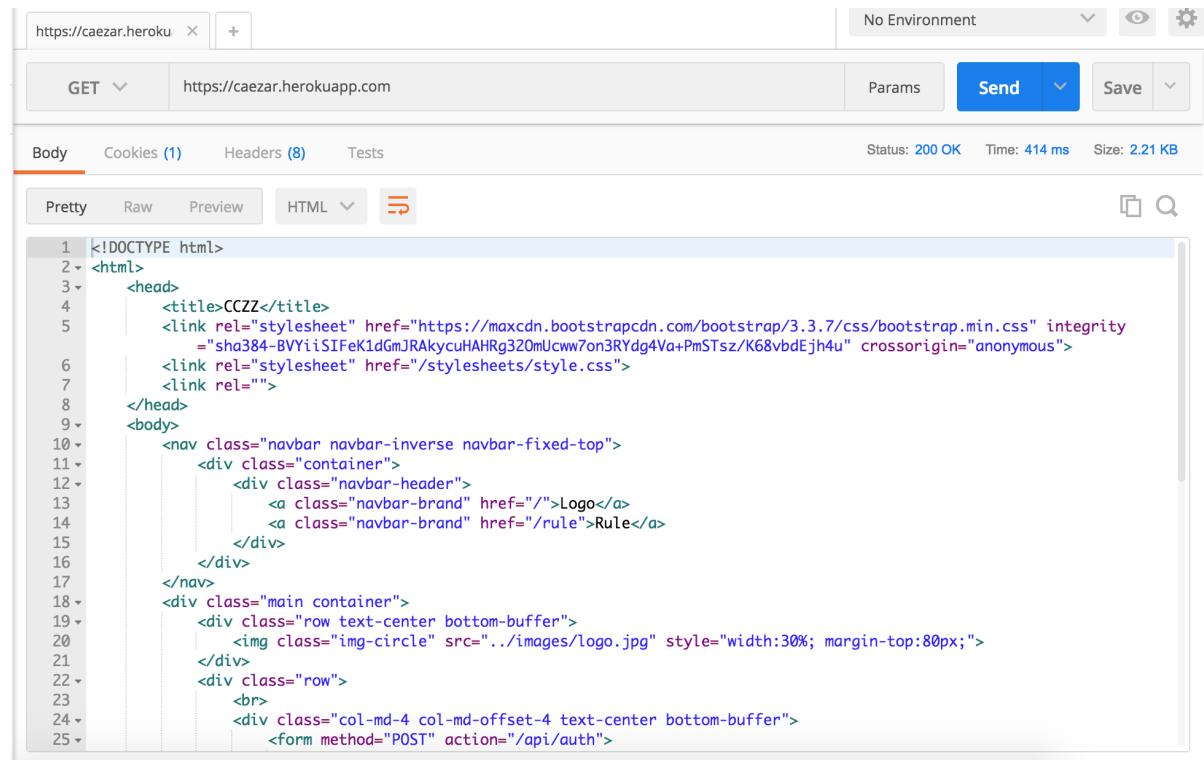
Category	Description	Total	Self Eval
<b>Code Quality</b>	Code is clean, well formatted (appropriate white space and indentation)	5	5
	Classes, methods, and variables are meaningfully named (no comments exist to explain functionality - the identifiers serve that purpose)	5	5
	Methods are small and serve a single purpose	3	3
	Code is well organized into a meaningful file structure	2	2
<b>Documentation</b>	A PDF is submitted that contains:	3	3
	Full names of team members	3	3
	A link to github repository	3	3
	A copy of this rubric with each item checked off that was completed (feel free to provide a suggested total you deserve based on completion)	1	1
	Brief description of architecture (pictures are handy here, but do not re-submit the pictures I provided)	5	5
	Problems you encountered during implementation, and how you solved them	5	5
	A discussion of what was difficult, and why	5	5
	A description of your test plan (if you can't prove that it works, you shouldn't get 100%)	5	5

Figure 6 Grade Rubric

We implemented login/logout, register, create a game, join a game, wait other players to begin the game, click play button to deal cards features. Although we don't have time to finish all the other functions, we learnt a lot from building this project! That is the most important thing to us.

## 6. Test Plan

Login:



The screenshot shows the Postman application interface. At the top, the URL bar displays "https://caezar.herokuapp.com". The main area shows a GET request to the same URL. The response status is 200 OK, with a time of 414 ms and a size of 2.21 KB. The response body is displayed in a code editor-like format, showing the HTML structure of the page. The code includes Bootstrap CSS imports and a navigation bar with links to 'Logo' and 'Rule'. A logo image is also present.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>CCZZ</title>
5     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRakycuHARg320mUcw7on3RYdg4Va+PmSTSz/K68vbEjh4u" crossorigin="anonymous">
6     <link rel="stylesheet" href="/stylesheets/style.css">
7     <link rel="">
8   </head>
9   <body>
10    <nav class="navbar navbar-inverse navbar-fixed-top">
11      <div class="container">
12        <div class="navbar-header">
13          <a class="navbar-brand" href="/">Logo</a>
14          <a class="navbar-brand" href="/rule">Rule</a>
15        </div>
16      </div>
17    </nav>
18    <div class="main container">
19      <div class="row text-center bottom-buffer">
20        
21      </div>
22      <div class="row">
23        <br>
24        <div class="col-md-4 col-md-offset-4 text-center bottom-buffer">
25          <form method="POST" action="/api/auth">
```

Figure 7 Test GET in Postman

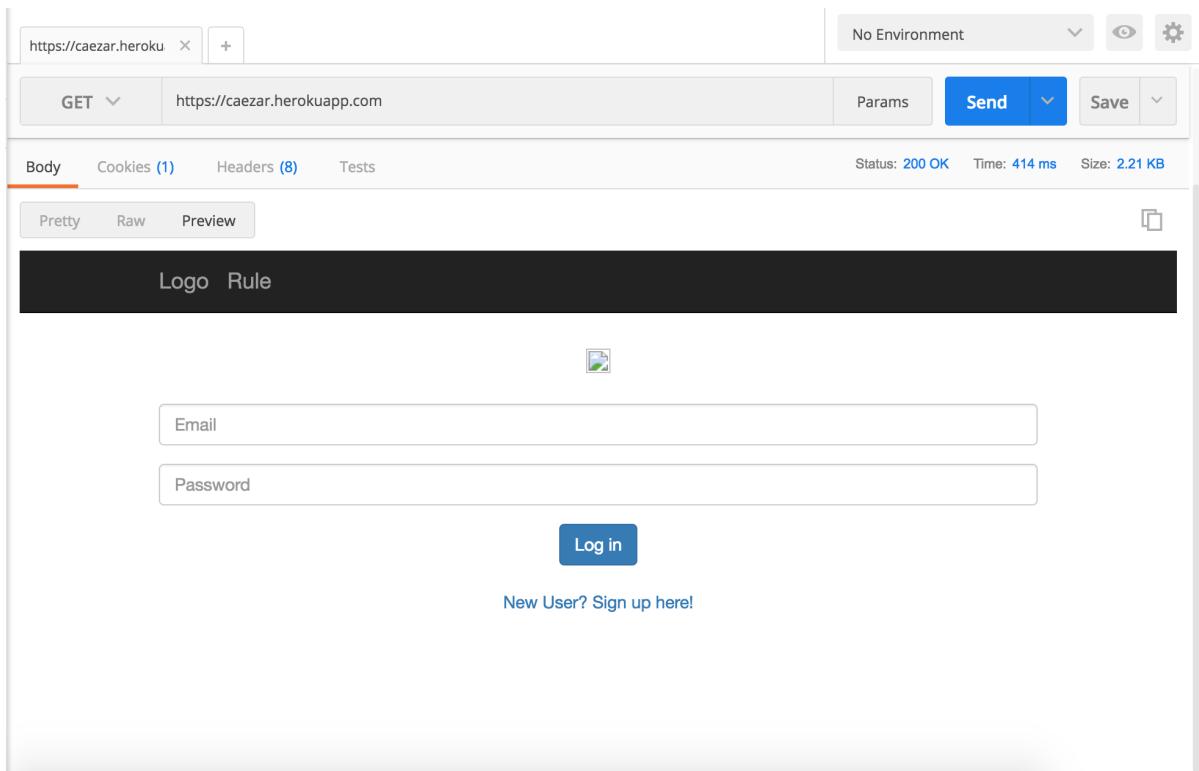


Figure 8 Test Login in Postman

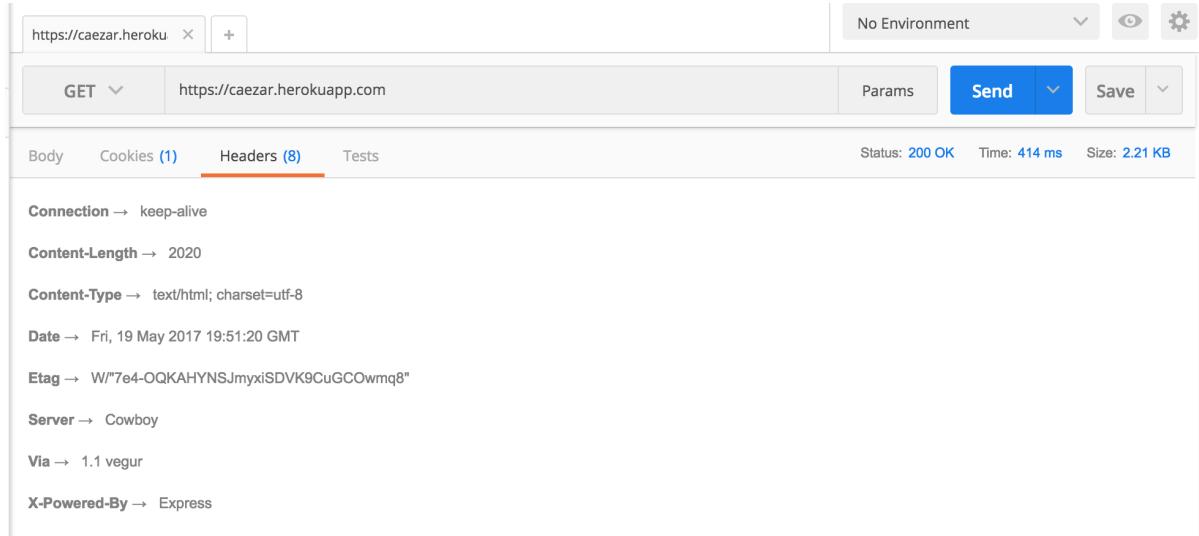


Figure 9 Get Response Back

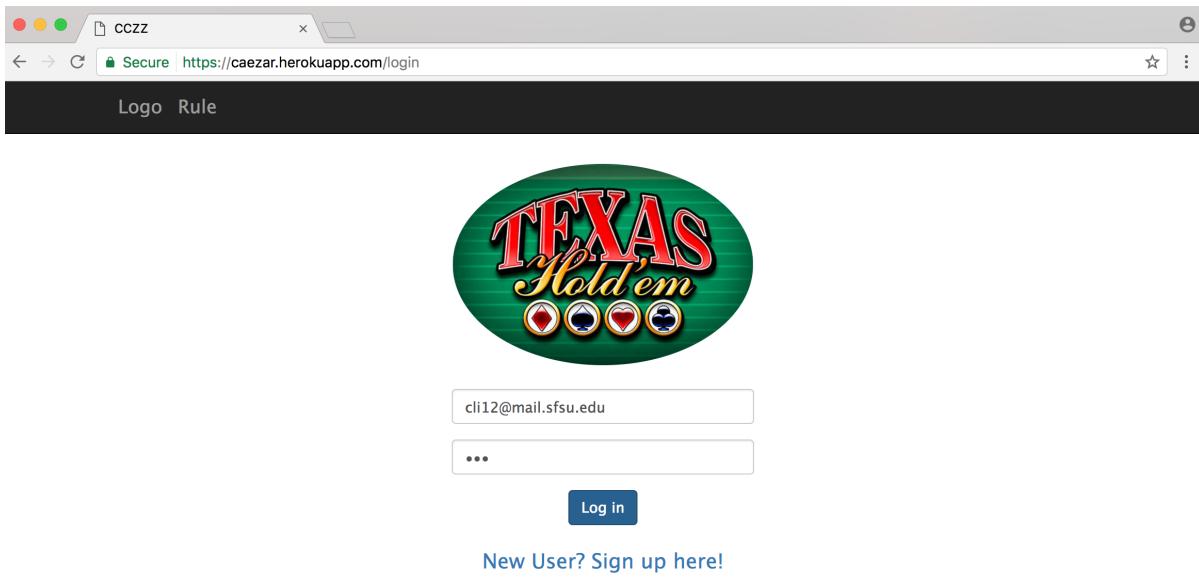


Figure 10 Login Interface

After Login:

The screenshot shows a web browser window titled 'cczz' with a secure connection to 'https://caezar.herokuapp.com/lobby'. The top navigation bar includes links for 'Logo', 'Rule', 'cli12', and 'Sign Out'. The main content area is divided into three sections: 'Score Board', 'Game Board', and 'Chat Board'. The 'Score Board' section displays JSON data for user 'cli12'. The 'Game Board' section shows room information for room 1. The 'Chat Board' section has a message input field and a 'Send' button. A 'Create New' button is located at the bottom of the lobby interface.

```

Score Board
cli12, email": "cli12@mail.sfsu.edu", "password": "123", "money": 100, "ranking": 1, "image_path": "./images/chip.png"

User: 2
{"id": 2, "username": "daydreamerlee", "email": "daydreamerlee@gmail.com", "password": "456", "money": 100, "ranking": 2, "image_path": "./images/chip.png"}

```

```

Game Board
Room: 1
{"id": 1, "dealer_pid": 1, "small_blind": 100, "player_amount": 1}

```

Figure 11 Lobby Interface

Register: After click the “New User” link on the login page:

The screenshot shows a web browser window with a registration form. The title bar says "cczz". The address bar shows "Secure https://caezar.herokuapp.com/register". The page has a black header with "Logo" and "Rule". Below it is a "Register" section with fields for "Username", "Email", "Password", and "Confirm Password". Each field has a corresponding input box. At the bottom are "Register" and "Reset" buttons.

Register

Username

Email

Password

Confirm Password

Register

Reset

Figure 12 Test Registration

Test user validation: if the confirm password is different from password:

The screenshot shows a web browser window with a registration form. A modal dialog box is open, displaying an error message: "caezaer.herokuapp.com says: The confirm Password you input is not the same as Password!". The "OK" button of the dialog is visible. Below the dialog, the registration form fields are shown: "Email" (test123@hotmail.com), "Password" (\*\*\*\*\*), and "Confirm Password" (\*\*\*). At the bottom are "Register" and "Reset" buttons.

caezaer.herokuapp.com says:

The confirm Password you input is not the same as Password!

OK

Email

test123@hotmail.com

Password

\*\*\*\*\*

Confirm Password

\*\*\*

Register

Reset

Figure 13 Test Registration Form Validation

There would be an alert window jumping out to tell the warning. And then if the user click OK, the page will stay on the register page.

After successfully register and login, the page will redirect to the lobby page.

The screenshot shows a web browser window with the title 'cczz'. The URL is https://caezaer.herokuapp.com/lobby. The page has a navigation bar with links for 'Logo', 'Rule', 'test123', 'Sign Out', and a dropdown menu. Below the navigation is a 'Score Board' section containing user data, a 'Game Board' section containing room data, and a 'Chat Board' section with a message input field and a 'Send' button.

User:	id	username	email	password	money	ranking	image_path
User: 1	1	cli12	cli12@mail.sfsu.edu	123	100	1	./images/chip.png
User: 2	2	daydreamerlee	daydreamerlee@gmail.com	456	100	2	./images/chip.png
	3	john	jrob@sfsu.edu	456	100	3	./images/chip.png
	4	test123	test123@hotmail.com	12345	100	0	
	5	max	max@gmail.com	123	100	0	

Room:	id	dealer_pid	small_blind	player_amount
Room: 1	1	1	100	1

**Score Board**

**Game Board**

**Chat Board**

Message...

Figure 14 After Registration Direct to Lobby

In Heroku database, the new user has been inserted to the user table: (new user: test123)

```
[caezaer::DATABASE=> select * from users;
+-----+-----+-----+-----+-----+
| id | username | email | password | money | ranking |
|-----+-----+-----+-----+-----+
| 1 | cli12 | cli12@mail.sfsu.edu | 123 | 100 | 1 | ./i
| 2 | daydreamerlee | daydreamerlee@gmail.com | 456 | 100 | 2 | ./i
| 3 | john | jrob@sfsu.edu | 456 | 100 | 3 | ./i
| 4 | test123 | test123@hotmail.com | 12345 | 100 | 0 | 
| 5 | max | max@gmail.com | 123 | 100 | 0 | 
(5 rows)

caezaer::DATABASE=> ]
```

Figure 15 Test Inserting New User in DB

Click the Rule:

The screenshot shows a web browser window with the title bar "cczz". The address bar indicates a secure connection to "https://caezaer.herokuapp.com/rule". The page header includes links for "Logo", "Rule", "test123", and "Sign Out". The main content area has a large heading "Rule" and a sub-section titled "Betting Structures". The text describes the rules of Hold 'em, mentioning blinds, dealer button rotation, and betting structures. It also notes that in two-player heads-up play, the dealer posts the small blind and the other player posts the big blind.

Hold 'em is normally played using small and big blind bets— forced bets by two players. Antes (forced contributions by all players) may be used in addition to blinds, particularly in later stages of tournament play. A dealer button is used to represent the player in the dealer position; the dealer button rotates clockwise after each hand, changing the position of the dealer and blinds. The small blind is posted by the player to the left of the dealer and is usually equal to half of the big blind. The big blind, posted by the player to the left of the small blind, is equal to the minimum bet. In tournament poker, the blind/ante structure periodically increases as the tournament progresses. After one round of betting is done, the next betting round will start by the person in the small blind.

When only two players remain, special 'head-to-head' or 'heads up' rules are enforced and the blinds are posted differently. In this case, the person with the dealer button posts the small blind, while his/her opponent places the big blind. The dealer acts first before the flop. After the flop, the dealer acts last and continues to do so for the remainder of the hand.

The three most common variations of hold 'em are limit hold 'em, no-limit hold 'em and pot-limit hold 'em. Limit hold 'em has historically been the most popular form of hold 'em found in casino live action games in the United States. In limit hold 'em, bets and raises during the first two rounds of betting are limited to the big blind; this amount is called the "ante" bet. In

Figure 16 Test Rule Page

After login, test the username link:

The screenshot shows a web browser window with the title bar "cczz". The address bar shows "localhost:3000/api/users/test123". The page header includes links for "Logo", "Rule", "test123", and "Sign Out". The main content area features a logo for "Texas Hold'em" with four playing card symbols (diamond, spade, heart, club). Below the logo, the user profile information is displayed: "User Name: test123", "Email Address: test123@hotmail.com", and "Money: 100".

User Name:  
test123

Email Address:  
test123@hotmail.com

Money:  
100

Figure 17 User Profile Interface

Test Sign Out in the navigation bar, the session will be destroyed and the page will be redirect to the login page.

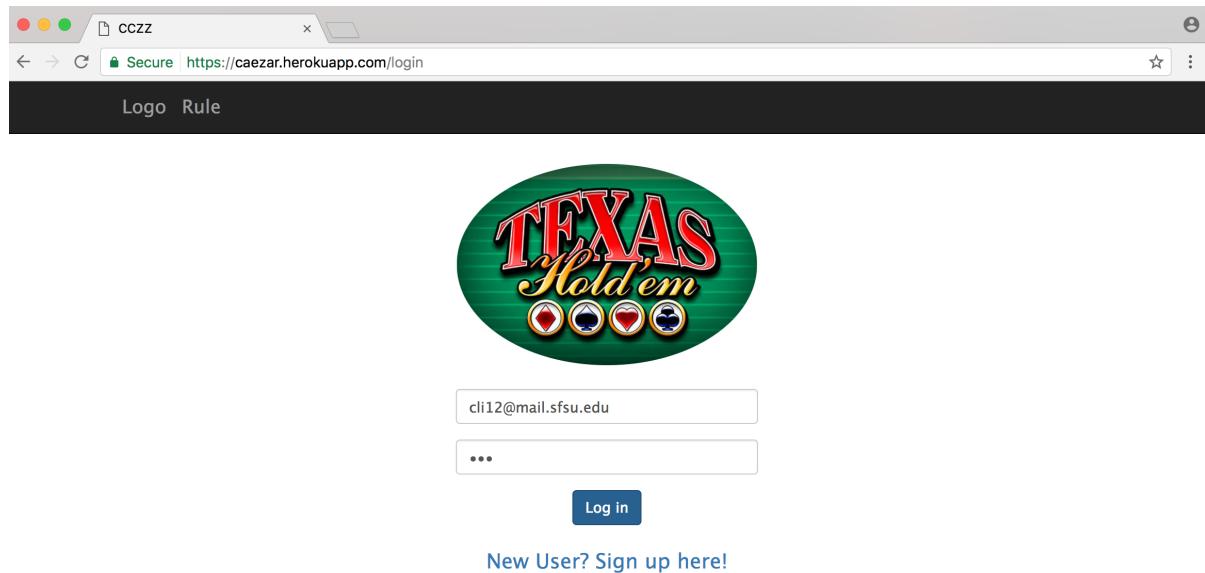


Figure 18 Interface After Signout

After joining the game room:

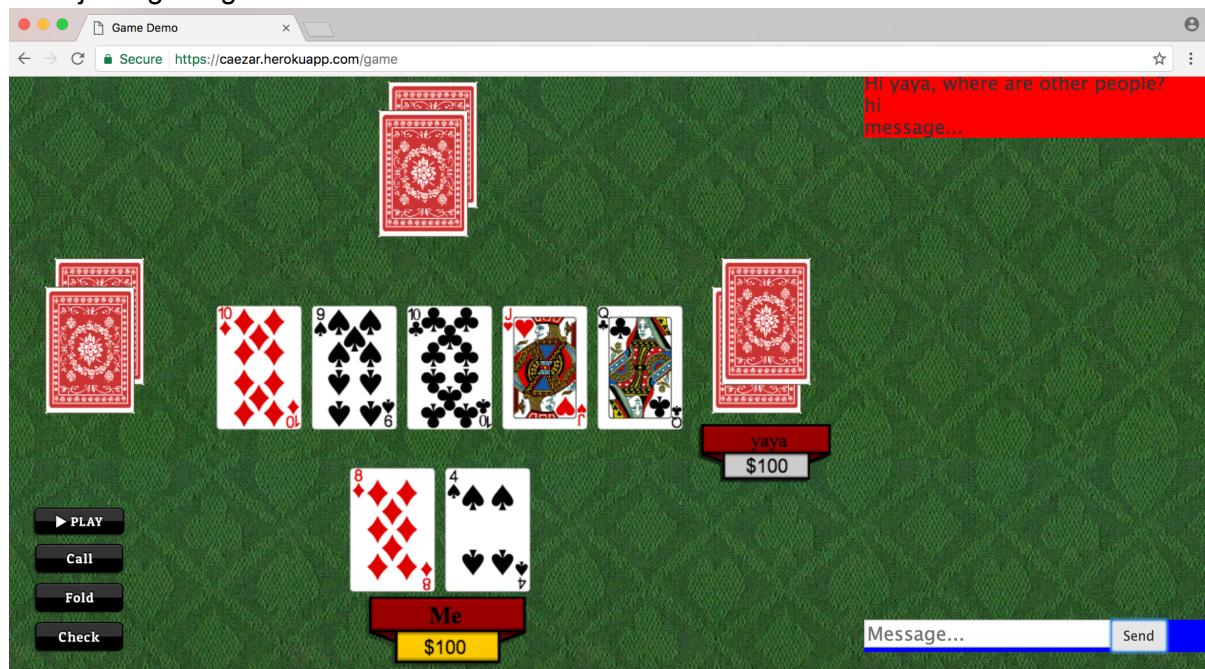


Figure 19 Game Page

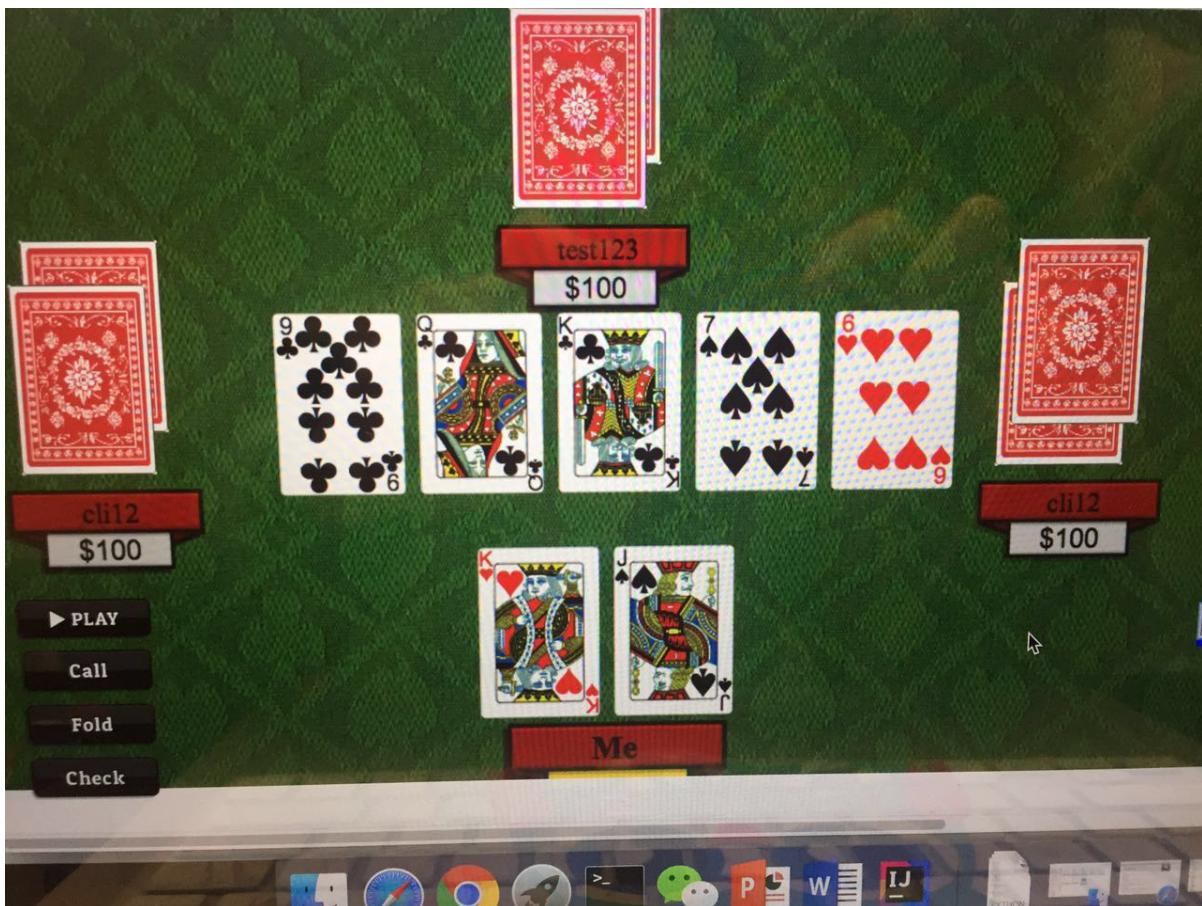


Figure 20 Play Game