

CAEZAR GAME

Team CCZZ

XINLU CHEN, CHENG LI, XUAN ZHANG, LIJIE ZHOU

OVERVIEW

1. Introduction
 - a. Project Overview
 - b. Database Diagram
2. Functionality and Main Challenge
 - a. Registration, Login/Logout (Xuan, Xinlu)
 - b. Lobby Page (Lijie)
 - c. Game Page (Cheng)
3. Conclusion and Future Works
4. DEMO

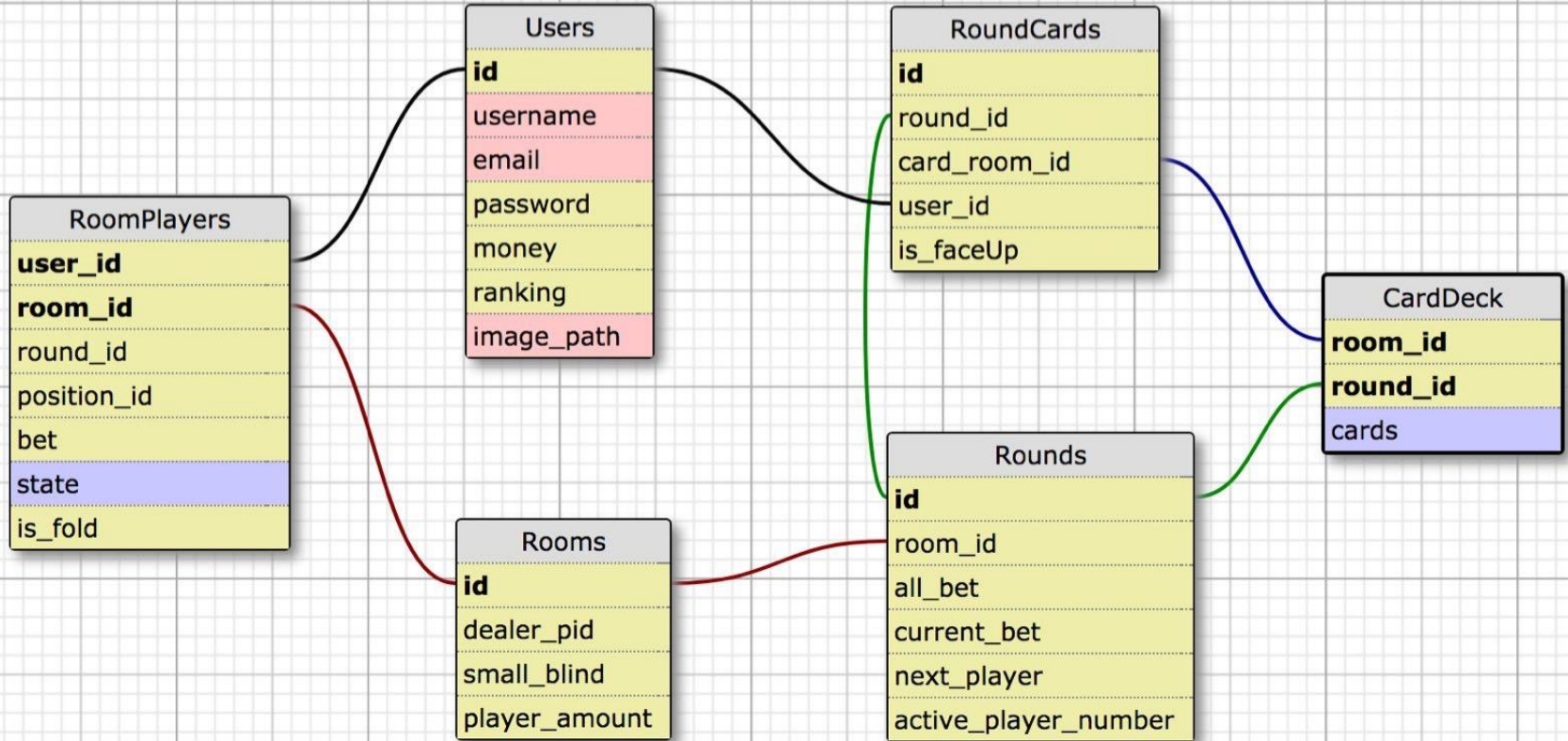
PROJECT OVERVIEW

- CaeZar is an online Texas Hold'em game.
- It supports multiple game rooms with up to 4 players in each game room.
- CaeZar allows chatting in the game lobby and in each game room.
- CaeZar provides game rules page for beginners as a reference.

PROJECT OVERVIEW

- Users need to register or log in to enter the game lobby.
- Users can create or join in a game room from the lobby to begin a game with friends.
- Users can view their profile after log in.

DATABASE DIAGRAM



REGISTER

CCZZ x

Secure <https://caezar.herokuapp.com/register> ☆ ⋮

[Logo](#) [Rule](#)

Register

Username

Email

Password

Confirm Password

[Register](#)

[Reset](#)

REGISTER

The screenshot shows a web browser window with the address bar displaying "https://caezar.herokuapp.com/register". The page has a dark header with "Logo" and "Rule" links. A modal dialog box is open, displaying the message: "caezar.herokuapp.com says: The confirm Password you input is not the same as Password!". Below the dialog, the registration form is visible. It includes a text input field with "test123", an "Email" label, an email input field with "test123@hotmail.com", a "Password" label, a password input field with four dots, a "Confirm Password" label, a confirm password input field with three dots, and two buttons: "Register" and "Reset".

CCZZ x

Secure https://caezar.herokuapp.com/register

Logo Rule

caezar.herokuapp.com says:

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

OK

test123

Email

test123@hotmail.com

Password

....

Confirm Password

...

Register


Reset

LOGIN/LOGOUT

CCZZ x

Secure https:// caezar.herokuapp.com/login

Logo Rule



Email

Password

Log in

[New User? Sign up here!](#)

LOGIN/LOGOUT

- Use session to check if a user logs in or not
 - If a user does not login, type <https://caezar.herokuapp.com/game> or <https://caezar.herokuapp.com/lobby>
 - will not lead to the corresponding pages.
 - will redirect to <https://caezar.herokuapp.com/login>

```
app.use('/', index);
app.use('/login', login);
app.use('/register', register);
app.use('/rule', rule);
app.use(function (req, res, next) {
  if (!req.session.user_id) {
    res.redirect('/login');
  } else {
    next();
  }
});
app.use('/lobby', lobby);
app.use('/game', game);
```

LOGIN/LOGOUT

- **Store user information in cookies when user login**
 - so that we don't have to send another request to server to get user information
- **Destroy session and clear cookies when a user clicks on Sign out button**

```
router.get('/', function (request, response) {  
  request.session.destroy(function() {  
    console.log("user logged out.")  
  });  
  response.clearCookie("email");  
  response.clearCookie("user_id");  
  response.clearCookie("username");  
  response.redirect('/login');  
});
```

LOBBY

1. Join room and create new room
2. \$ get
3. Chat board

← → ↻ 🏠 Secure https://caezar.herokuapp.com/lobby 🔑 ☆ 🛠️ 🗑️ 🚫 ⋮

📱 Apps 🌐 Login - San Francis... 📁 Hacker Rank 📁 CodePath 📁 Hack 📘 (2) Facebook Caree... 📧 Refuge Newsletter... 📧 Spring 2017 - CSC... 📁 Thesis >>

Logo Rule john Sign Out

Score Board

User: 1

```
{ "id": 1, "username": "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": 1, "image_path": "./images/chip.png" }
```

Game Board

Room: 1

```
{ "id": 1, "dealer_pid": 1, "small_blind": 100, "player_amount": 1 }
```

Room: 2

```
{ "id": 2, "dealer_pid": 1, "small_blind": 100, "player_amount": 1 }
```

Chat Board

hello
how are you?

🗑️

Send

Create New

LOBBY

1. Join room and create new room: use cookie to get user_id and room_id
2. All the data populated on this page is by using JQuery's \$.get method.

```
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      });
```

1. Chat board: using socket to handle

Client-side:

```
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      });
```

LOBBY

Server-side socket

```
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      });
```

GAME

- Canvas
 - Name plate
 - Cards
 - Bet amount
- Form/submit
 - Action buttons



GAME

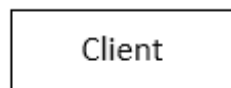
1. socket.io to handle the Client /Server communication (user-return, user-join, start-play, room-chat)

1. Use socket.io /room

a. Socket.join

b. io.sockets.in(room).emit(...)

USER_JOINED



```
socket.emit(USER_JOINED, {userid: userid,  
roomid: roomid, username: username});
```

Socket (roomid)

Other client (user) in that room

Server

```
socket.on(USER_JOINED, function (data) {  
  var userid = data.userid;  
  var roomid = data.roomid;  
  var username = data.username;  
  
  socket.join(data.roomid);  
  
  db.getRoomById(data.roomid).then(function (data) {  
    db.updateRoomById(data.id, data.player_amount + 1).then(function (data) {  
      db.createRoomPlayers(userid, data.id, username, data.player_amount).then(function (data) {  
        io.sockets.in(roomid).emit(USER_JOINED, {roomid: roomid, userid: userid, positionid: data});  
      });  
    });  
  });  
});
```

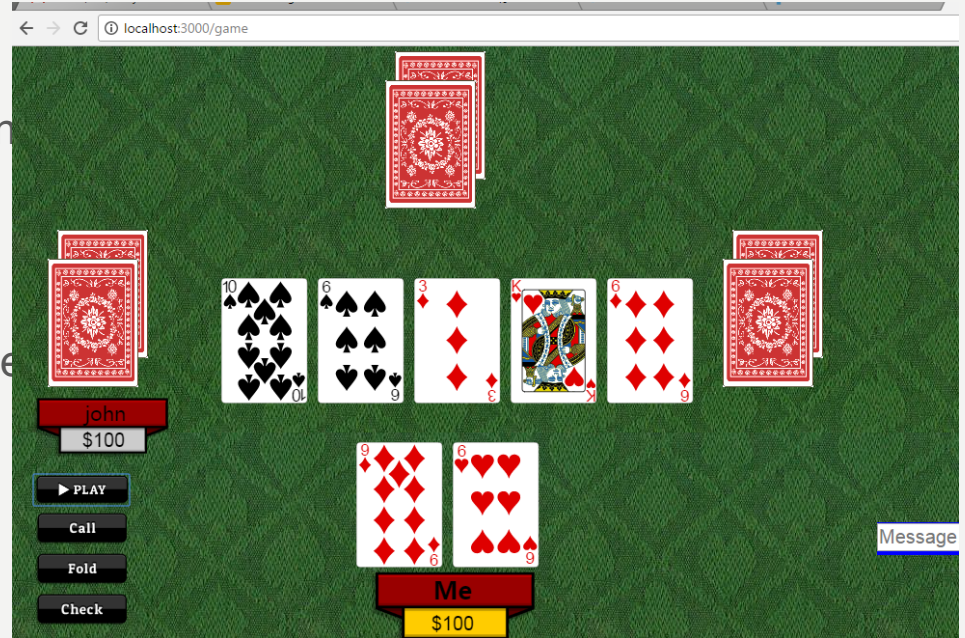

GAME

Canvas

- Name plate (user_join/user return)
- Cards (start_play)
- Bet amount (update_game)

Form/submit

- Action buttons (trigger start_game)
- update_game)



CONCLUSION

- Features Implemented

login/logout, register, user profile, chat, create/join a game, wait other players to begin the game, click play button to deal cards.

- Although we don't have time to finish all the other functions, we learnt a lot from building this project! That is the most invaluable experience to us.

FUTURE WORKS

We would like to finish other features and make this game work!

DEMO

Thanks for your time!