

Blackjack

Project 1 – Web Game

Team:

Darshit Chanpura: 001854701

Abhishek Karan: 001883648

Game Description:

Blackjack is a casino game. The Blackjack, also known as 21, has been one of the most popular North American casino games through the last hundred years and has spread throughout the world. The aim of blackjack is to beat the dealer. To beat the dealer the player must be wary of the following: 1) do not "Bust" (go over 21), 2) either outscore the dealer or have the dealer bust.

Equipment:

Blackjack is played with an international, standard deck of cards with the Joker cards removed, leaving 52 cards. Originally the game was played with a single deck, so in this project we are following this convention. However, as a counter measure to card counting, casinos introduced multi-deck games, based on the false assumption that if there were more cards in play it would be harder for the card counter to keep track of them all. So, to counter this assumption we are not showing the cards dealt once the other players have been "Busted" or clicked "STAY" for their turn.

General Rules:

- There can be 7 players playing along with dealer on a single table.
- Aces are counted as 1, 2 to 9 are the pipe values. 10, Jack, Queen & King are of 10 points each.
- Play begins with the player to the dealer's left. The following are the choices available to the player:
 1. **Stand:** Player stands pat with his cards.
 2. **Hit:** Player draws another card (and more, by calling Hit again, if he wishes). If this card causes the player's total points to exceed 21 (known as "breaking" or "busting") then he loses. And if the total points reach 21, then he wins.
 3. **Surrender:** The player forfeits his chance and plays out his turn. This option is available during anytime of the game. This is a mandatory action when you want to leave the table.

- Every player gets a chance to play his/her hand after the previous player either gets “Busted” or has stood his chance or surrenders the game.
- On surrendering the chance, the player loses his turn. Any new player may come and join at the same spot.

Customized Rules:

- Aces have a face value of 1 instead of 11 or 1.
- There is no dealer against whom the player is playing.
- The players are not playing against the Dealer but instead with other players for a more engaging game play experience.
- Minimum of 2 players are required to start the game play.
- No bets of any sort are allowed. We want to keep the game as legal as possible!
- There is no splitting of cards with face values of 10.
- There are no Joker cards in the game.
- If Ace is dealt then, no insurance is given to any player for saving against being “Busted”.
- Joining as a “Spectator” is allowed at any time. However, the spectator can only view the game and then go out by clicking the Surrender button.
- When the table has reached its maximum capacity of player count i.e. 7 then the player who joins will join as spectator.

Gameplay Options:

- A player has 2 options (HIT/STAY) to toggle on every chance, with an open option to SURRENDER the game at any stage. These options come in form of 3 buttons respectively.
- There is token system running throughout the game which informs server as to which player's chance it is.
- Once the player clicks on the “HIT” button, a single new card is dealt to him in front of his already dealt cards, if any, which comes from a shuffled deck of cards from server.
- If the player clicks the “STAY” button then the token moves to the next player. The current state of this player remains the same.
- None of the other players will get the same card as the others.

Game Setup:

- In the Lobby, the player has the option to choose any of the table of his/her choice.
- It has an option to create a new Table as well.
- Each Table can accommodate a maximum of 7 players.
- If the player count of a particular-table reaches 7 then no more players are allowed to join the game as players. However, an opportunity is given for the person to view the ongoing game as a spectator/observer.
- A minimum of 2 players are needed to start a game on a particular table.
- The dealer is just a dead object who deals the cards and is not a part of the actual game experience.

Cards Setup:

- There is a deck of 52 regular casino cards.
- Cards are a combination of 4 suits (Heart, Diamond, Clubs, Spades) and of 2 colors i.e. Red & Black.
- A card from well shuffled deck is dealt to each of the player as and when the HIT button is pressed.

Technical Stack:

Front End (UI)

- ReactJS is used to render the components on the client side.
- Bootstrap and jQuery are used to beautify the front end.
- Backgrounds and Tables are shown using wide high-quality images.
- Cards dealt are shown using a deck of 52 images.
- Design Strategies:
 - “CardsContainer” component is used to render the cards dealt on the table.
 - Peripherals like the “Player Name” & “Score” are rendered on a single child div of the react root component.
 - The current table events are shown on a div on the right hand side of the screen. This tracks all the status updates of the table for a better picture.
 - This is also useful for the observer to know what is happening on table being observed.

Back End:

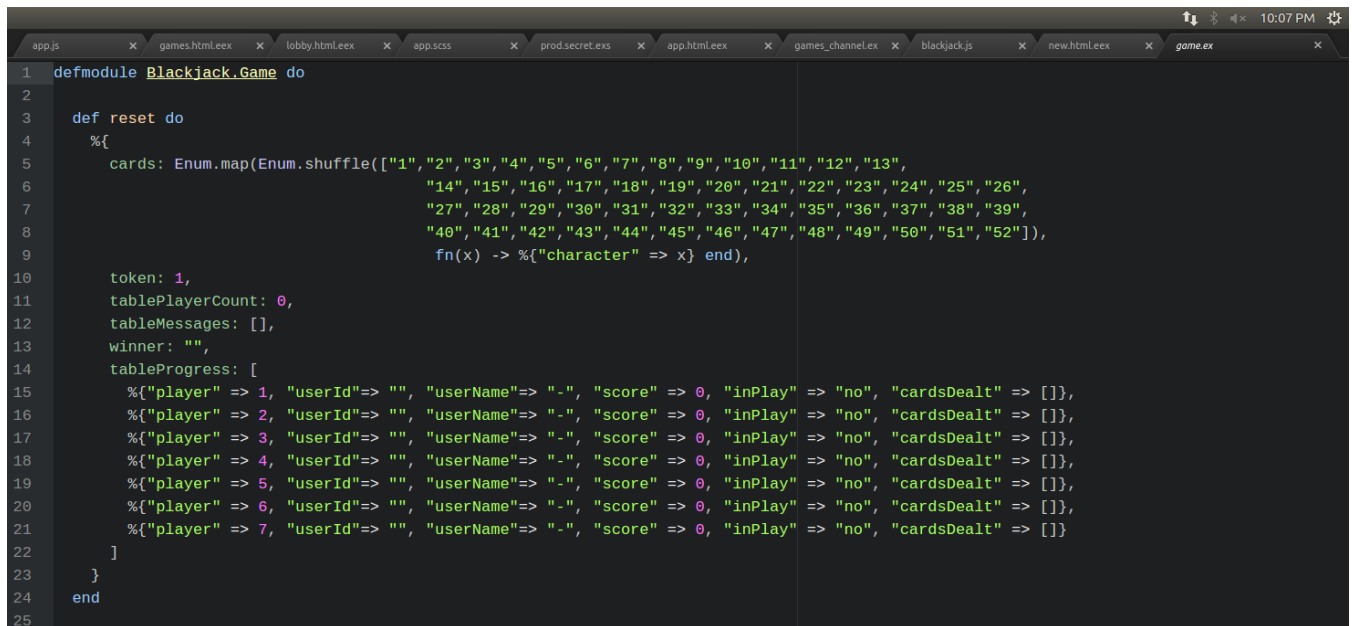
- The server side is coded using Elixir.
- This handles the cards shuffling and state updation of the table which in turn is the state of the game in hand.
- We have an Agent Server which holds the backup for the game state.
- Players join the table through web sockets and channels are then used to create a secure link for the same.
- Sessions are also maintained for each logged in player.
- Each table is a separate topic on the “game” channel.
- Each table gets a separate set of game state and thus a different set of shuffled cards for each player.

Database:

- We have used Postgres database.
- This is used to store the registered information of the user and the tables.

Game State (Server Side):

The entire game state progress is maintained through the game state variable at the server side. The table state basically becomes our game state. As shown in the figure [] below.

A screenshot of a code editor with multiple tabs at the top: app.js, games.html.ex, lobby.html.ex, app.scss, prod.secret.ex, app.html.ex, games_channel.ex, blackjack.js, new.html.ex, and game.ex. The active tab is game.ex. The code is in Elixir and defines the Blackjack.Game module. It includes a reset function that initializes the game state with a shuffled deck of 52 cards, a token of 1, a player count of 0, an empty messages list, a winner string, and a table progress list for 7 players. Each player's progress is initialized with empty fields for user ID, name, score, in-play status, and cards dealt.

```
1 defmodule Blackjack.Game do
2
3   def reset do
4     %{
5       cards: Enum.map(Enum.shuffle(["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13",
6       "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26",
7       "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39",
8       "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52"])),
9       fn(x) -> %{character => x} end),
10      token: 1,
11      tablePlayerCount: 0,
12      tableMessages: [],
13      winner: "",
14      tableProgress: [
15        %{player => 1, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
16        %{player => 2, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
17        %{player => 3, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
18        %{player => 4, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
19        %{player => 5, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
20        %{player => 6, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []},
21        %{player => 7, "userId"> "", "userName"> "-", "score" => 0, "inPlay" => "no", "cardsDealt" => []}
22      ]
23    }
24  end
25 end
```

Figure 1: Game State

This consists of the following:

1. Cards: This has the entire card map from 1 to 52. These number are then being mapped by the images stored in the static folder.
2. Player Count: This keeps track of the number of players in current table.
3. Token: This has the current token value which is mapped to the current player's turn.
4. Winner: This has the player name who won the game.
5. Table Progress: This has each of the player progress on the current table. It has values like Player Number, User ID, User Name, Cards Dealt (this has all the cards being dealt to the particular player), Score, In Play (this states that the player is active on the table and not has been busted or has not surrendered the game.)
6. Table Messages: This has the messages log from the table events like HIT, STAY, Join, Win from each user to be displayed at the table.

Game Features:

- Virtually unlimited number of players can play the game.
- Virtually unlimited number of tables can be created.
- Only 7 players are allowed at max on each table.
- Virtually unlimited number of spectators can watch the game progress.
- Each table has a Chat feature as well through which the players can communicate with each other.
- Each event like HIT, STAY, SURRENDER, JOIN etc are logged and displayed on the table as well.
- Score and Player Name of each player are shown near their seats.
- Every user has a session maintained to login and logout from the game.

Game Flow Chart:

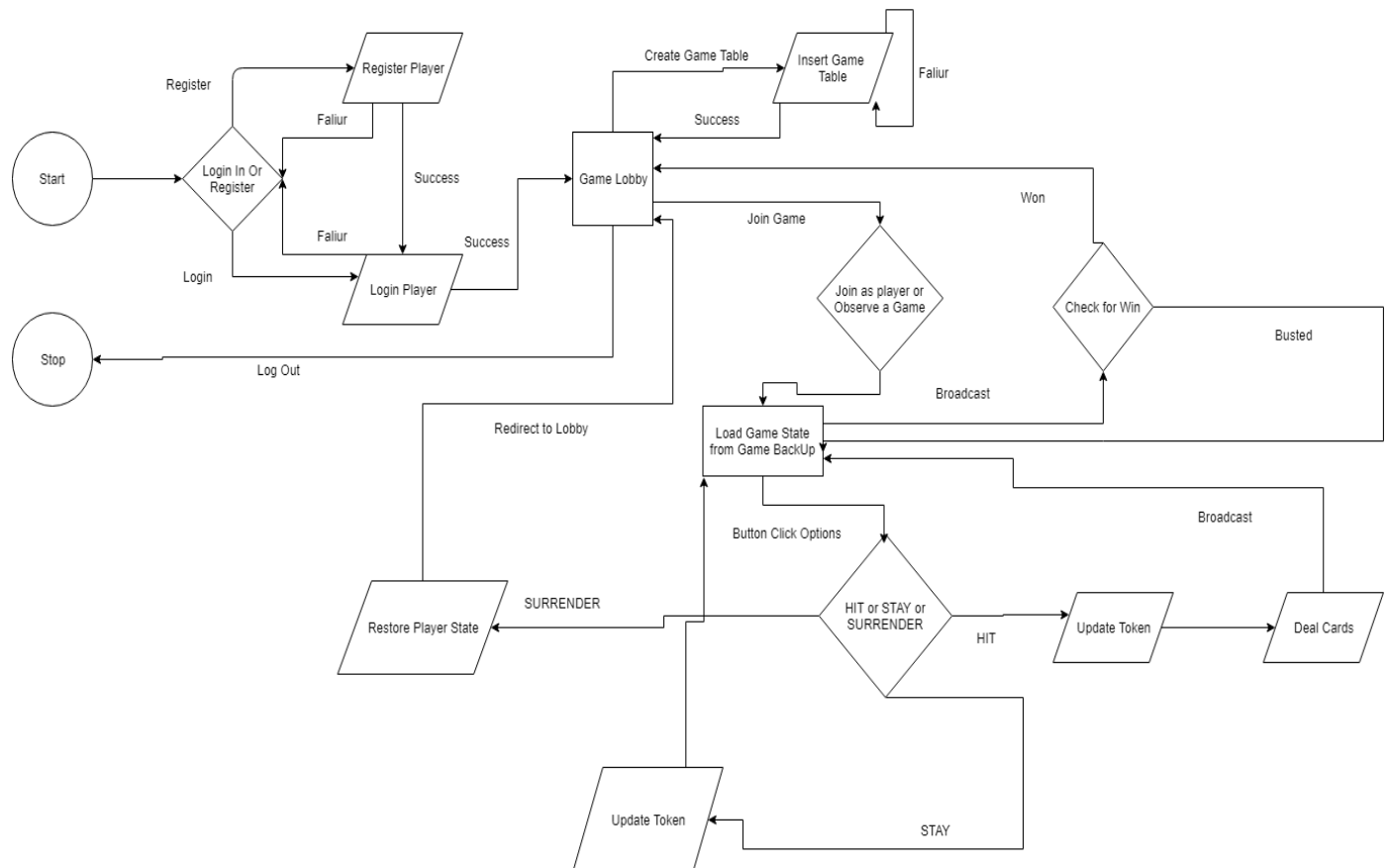


Figure 2:Flow chart

As shown in the figure above the game starts by the user either **Logging in** or can **register**. On successful sign in the user is directed to the **lobby**. Here the user has an option to create a

table or join an existing table. There are further couple of more options to join the game. These are in terms of actual players or as a spectator. There are different roles of each of them. The player plays against other players on the table in real time. Also, the players as well as spectators has the privilege to chat with other players and spectators. The spectator however can view the table progress and chat with the player playing on that table.

The game only starts when there are at least 2 players on the table. Not including the spectators.

During the gameplay the player has an option to HIT, SURRENDER or STAY his/her chance. On HIT, a card is dealt from a set of shuffled deck. The value gets added to the score. On STAY the token is updated to the next user on the table. On SURRENDER, the game state is reset for the next game and the player is redirected the lobby.

A player wins the game when, he/she scores 21 or when all the other players have all been busted. To further enrich the gameplay experience the player has the option for a real time chat and real time table updates as well.

Challenges & Solutions:

- Broadcasting the game state to each user so that everyone can see the game in real time was a major issue. We solved this issue by adding the following code to the `games_channel.ex` file.
- Rendering each card for the user dynamically on the HIT button click was difficult on React portion of the game.
- Chat Room implementation: Broadcasting all the chats as and when happens in real time created problems initially. However, through the use of the ``this.channel.on()`` function we were able to resolve it.
- UI challenges. The cards dealt by each user had to be shown in front of the respective user. Aligning chats and game logs was troublesome.
- Making responsive pages for rich mobile experience also took a lot of time and effort.

Game Sequence Diagram:

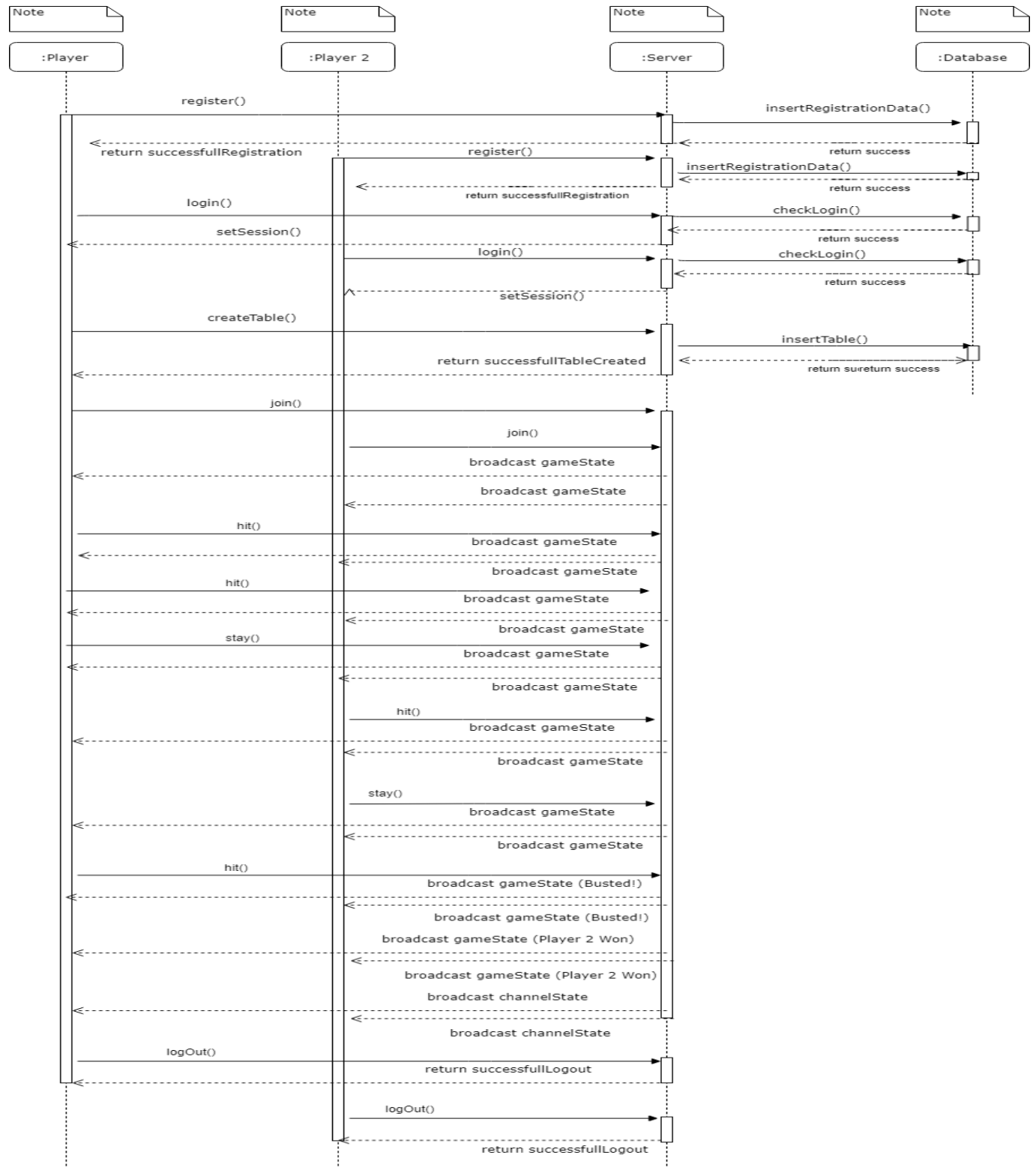


Figure 3: Game Sequence Diagram

The diagram above explains the game flow between two players with server and database. However, the order of player joining the table does not matter. The game is fair to each and every user. Once the session for each user is created and the tables are also created then there is no database interaction going to happen. All the game state is handled by the Phoenix sever.

Game Back-Up:

Agent server is used for storing the game back up. This gives the application a simple abstraction around the state. The Agent module provides a basic server implementation that allows state to be retrieved and updated via a simple API.

Though the use of Agent module, a seamless gameplay experience is seen. If the user mistakenly closes the browser then this module helps to restore the backed-up state from the server for the specific user. The state is updated through the `this.channel.on()` function. Every state is fetched from the backed-up state on the server and then broadcasted so that every user is on the same game play.

Screenshots:

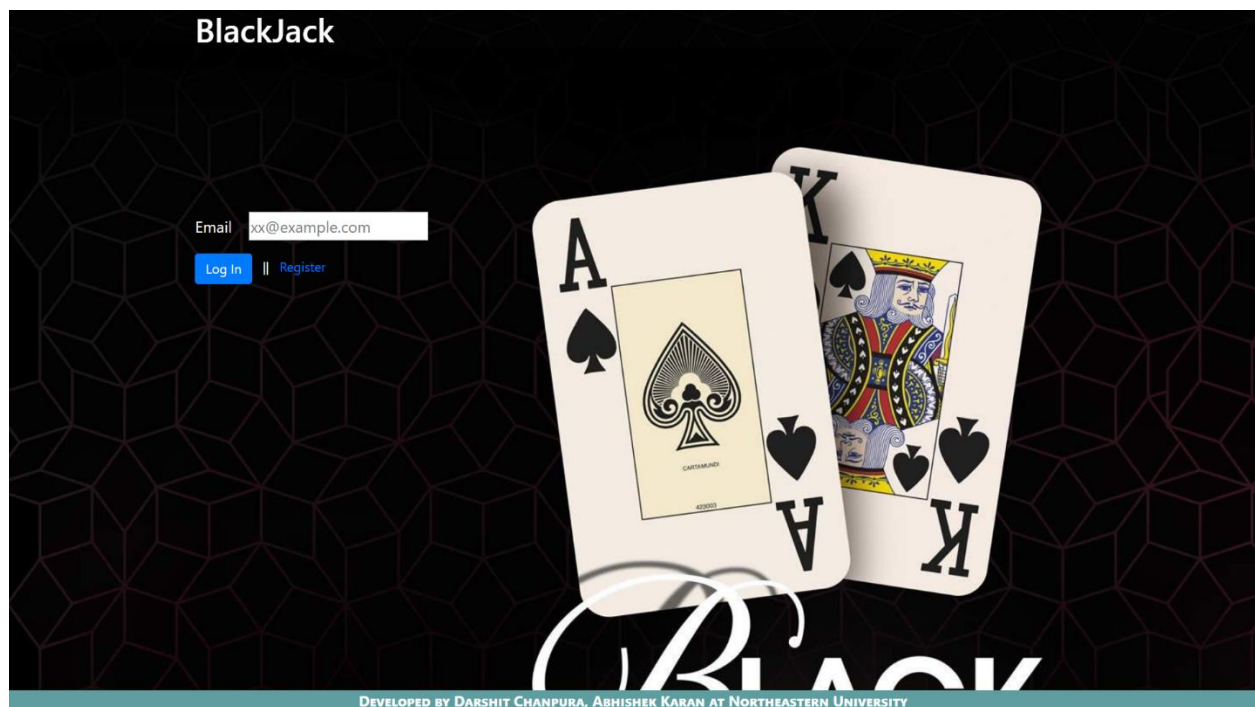
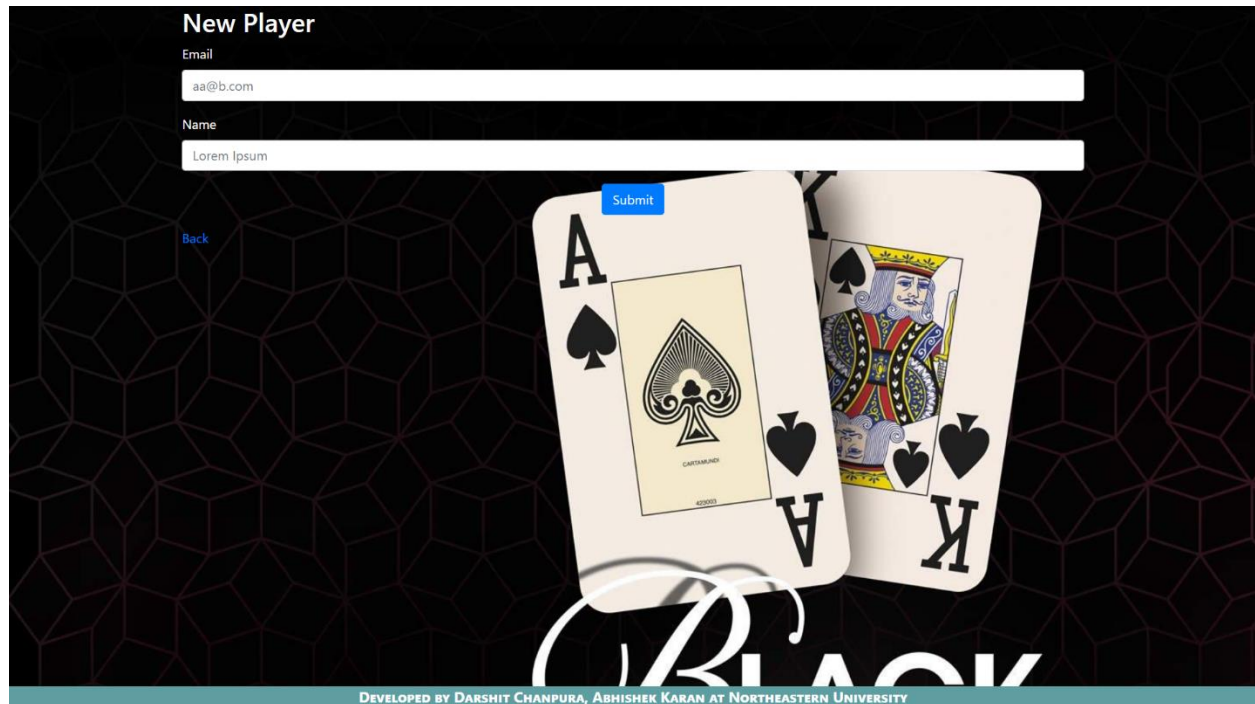


Figure 4: Login Page



The image shows a registration form titled "New Player" on a dark background with a subtle geometric pattern. The form includes two input fields: "Email" with the placeholder "aa@b.com" and "Name" with the placeholder "Lorem Ipsum". A blue "Submit" button is positioned above the input fields, and a blue "Back" link is to the left. The background features two playing cards, the Ace of Spades and the King of Spades, and the word "BLACK" in large, stylized white letters. A teal footer bar at the bottom contains the text "DEVELOPED BY DARSHIT CHANPURA, ABHISHEK KARAN AT NORTHEASTERN UNIVERSITY".

New Player

Email

aa@b.com

Name

Lorem Ipsum

Submit

Back

BLACK

DEVELOPED BY DARSHIT CHANPURA, ABHISHEK KARAN AT NORTHEASTERN UNIVERSITY

Figure 5: Register User

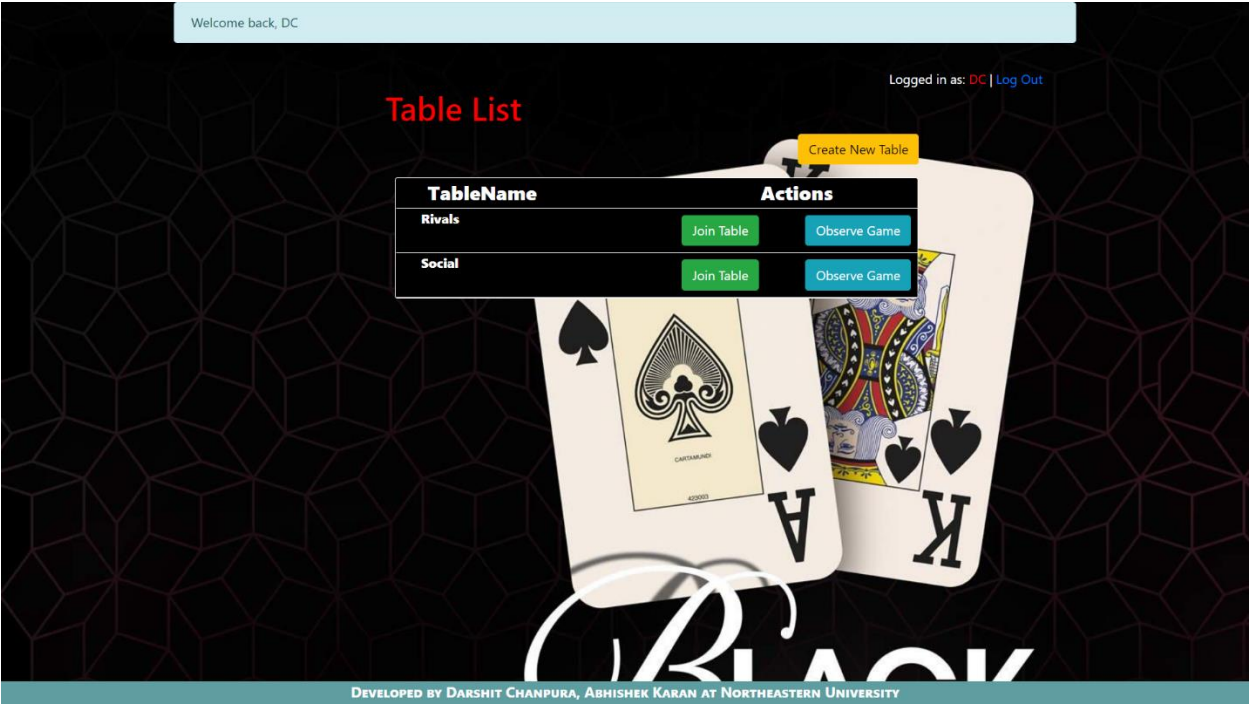


Figure 6: Lobby Page

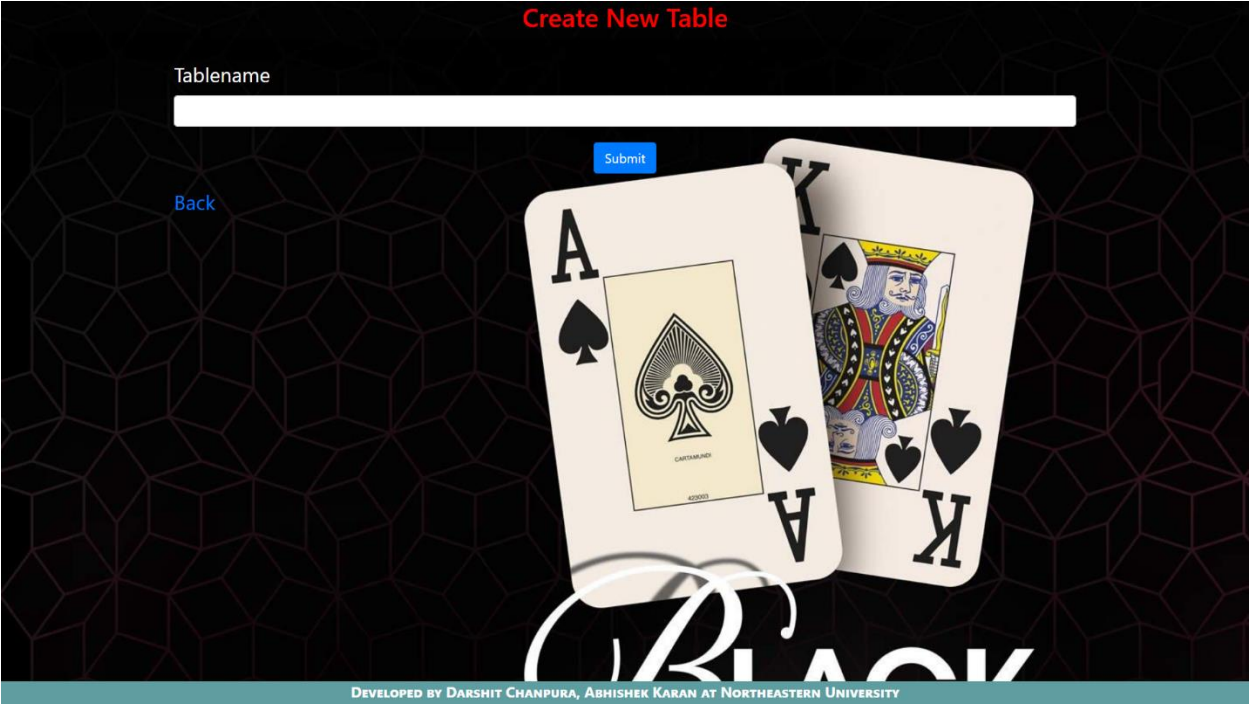


Figure 7: Create Table

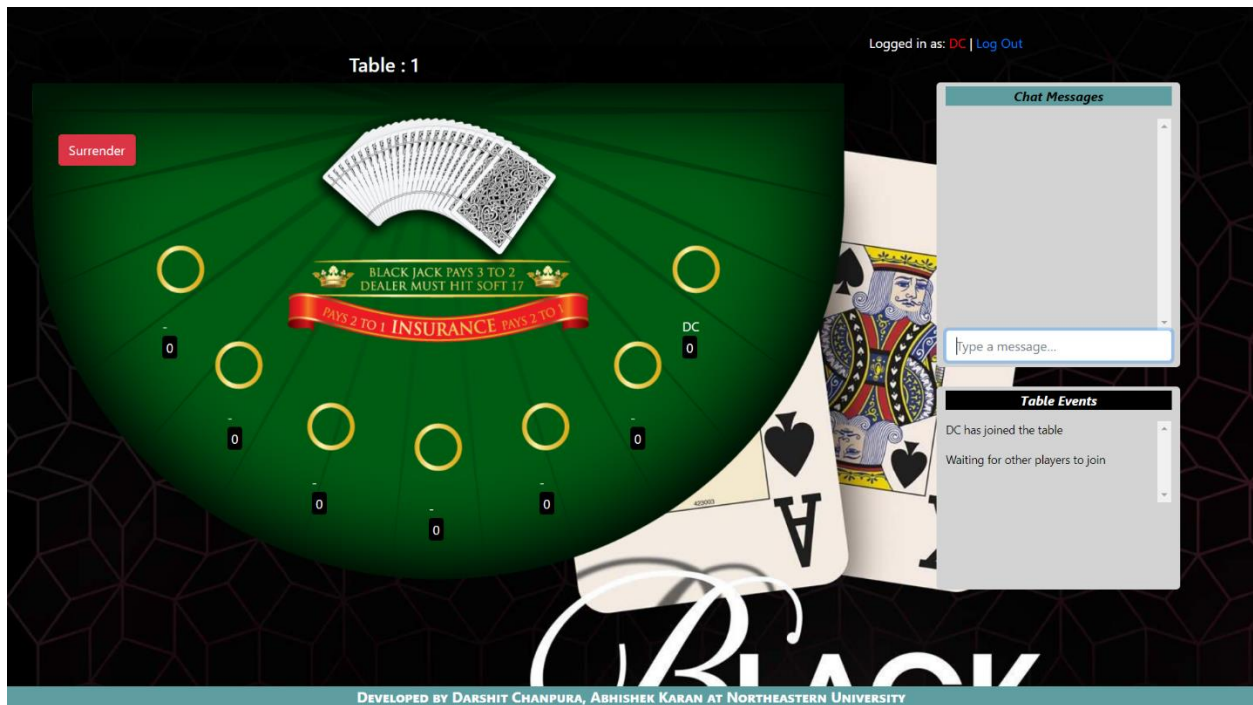


Figure 8: : Initial Game State as seen by first player

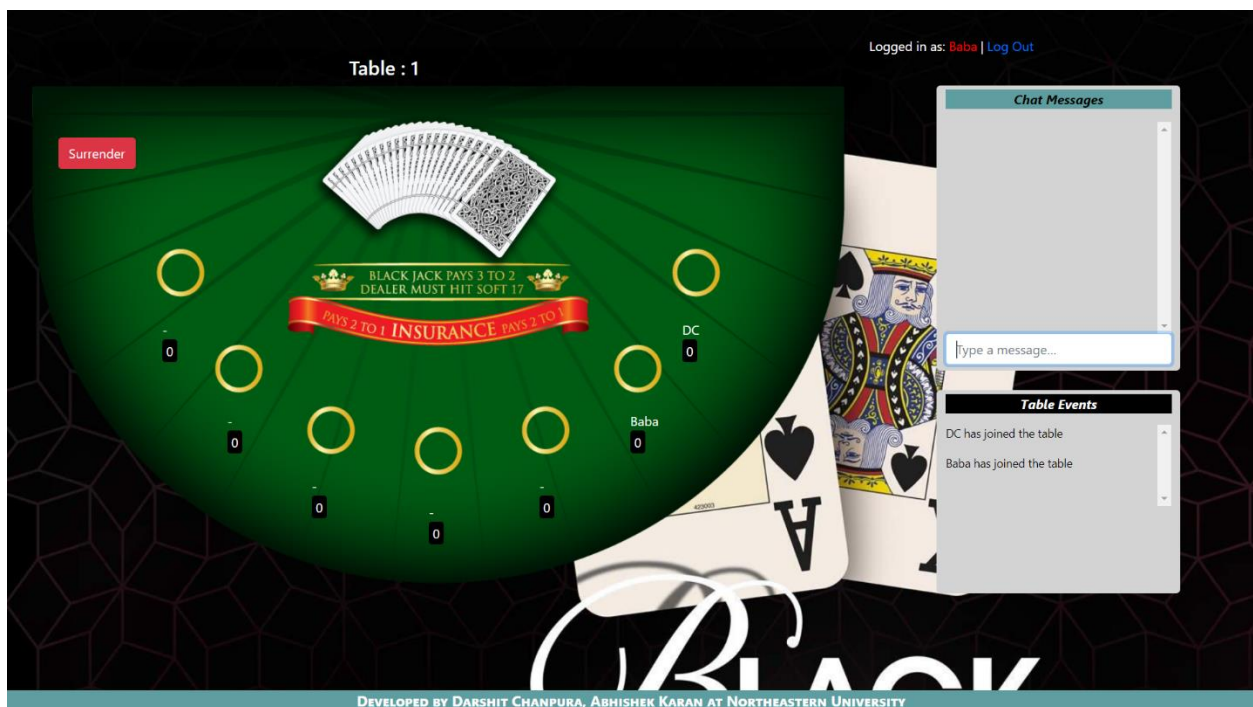


Figure 9: Initial Game State as seen by second player

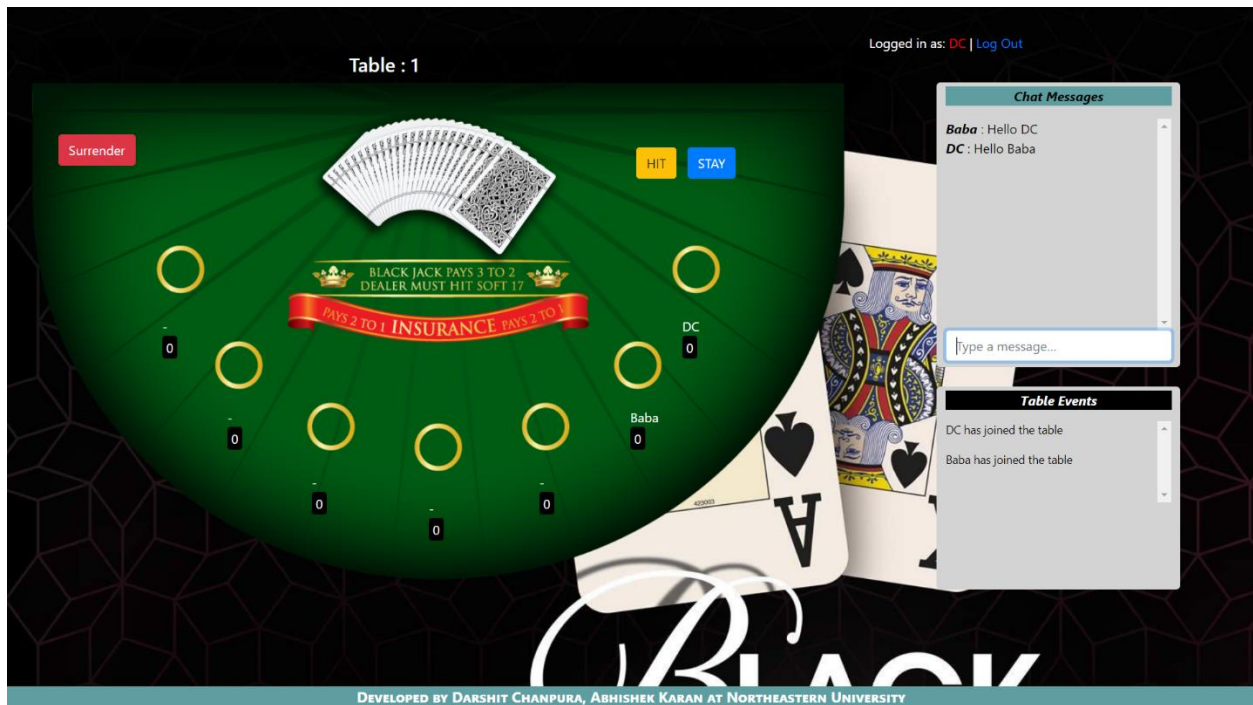


Figure 10: Players Chatting

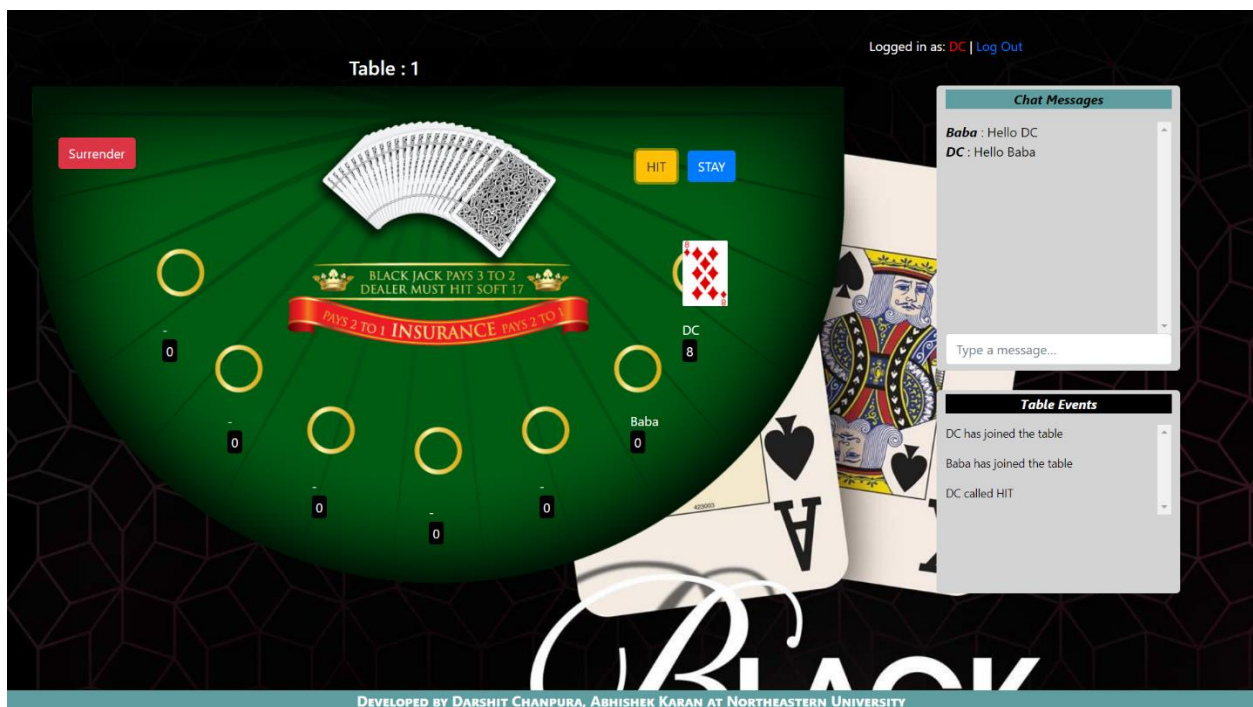


Figure 11: Player One calls HIT

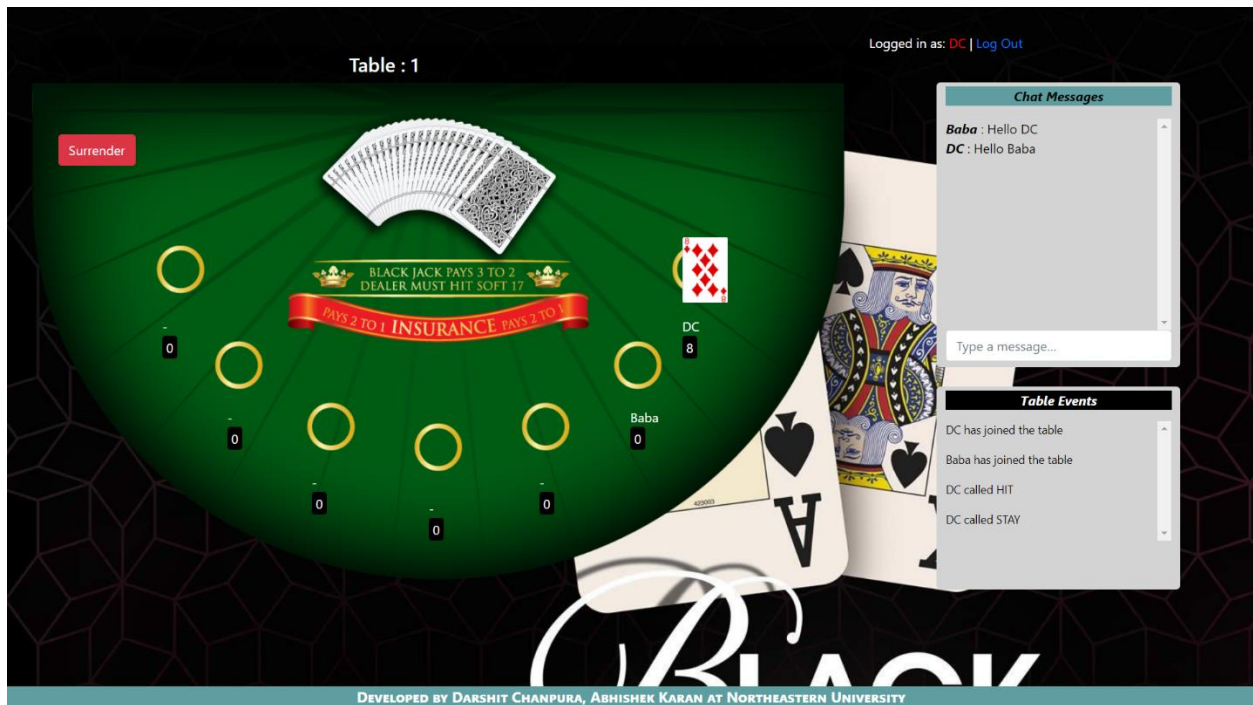


Figure 12: Player One calls STAY

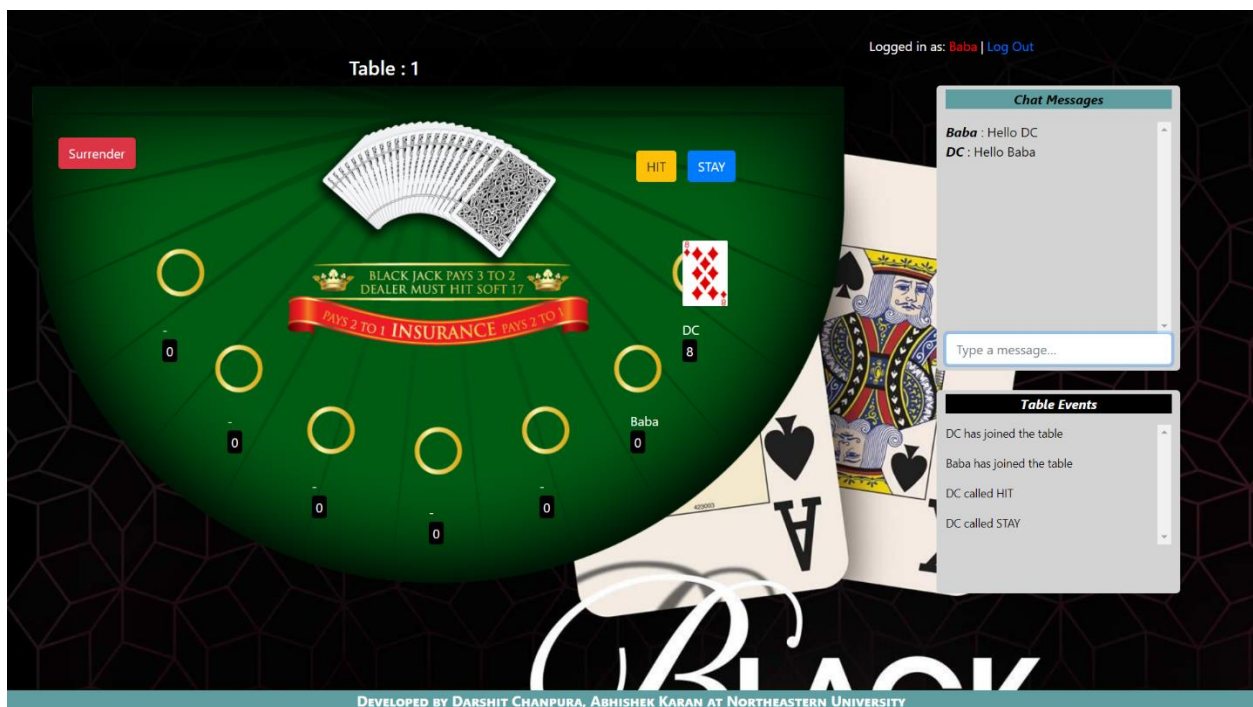


Figure 13: Second Player's turn

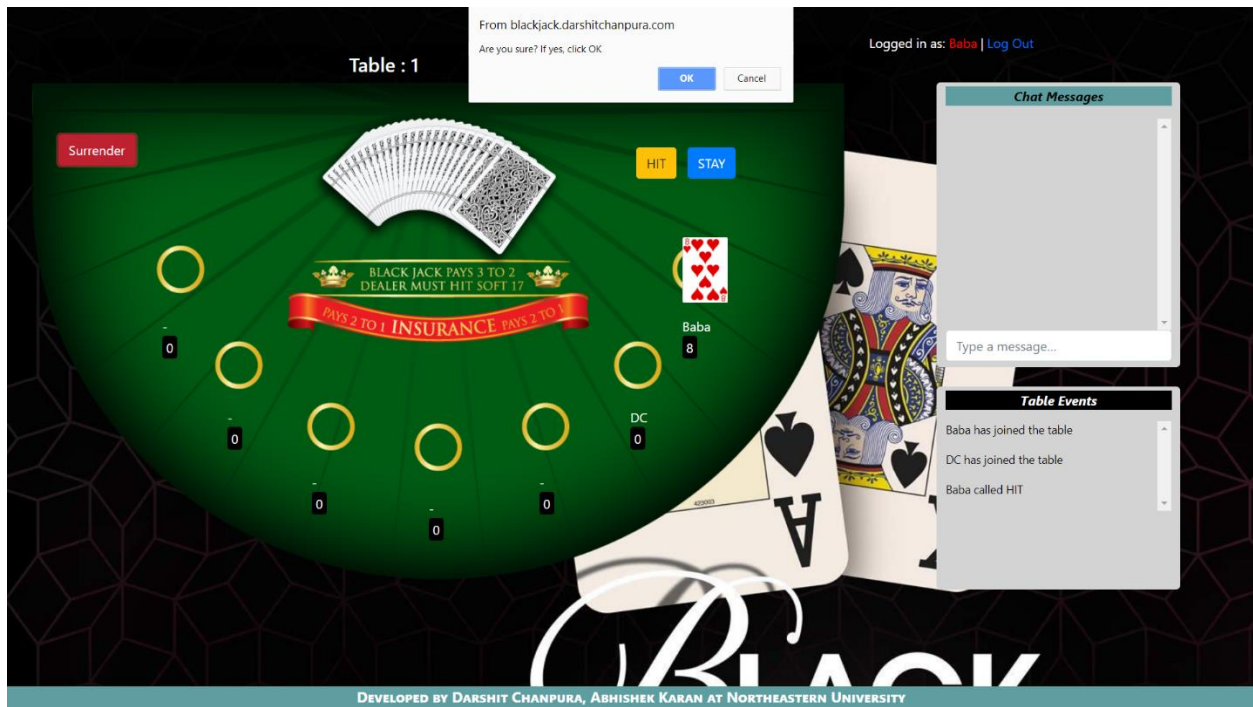


Figure 14: Player One Surrenders

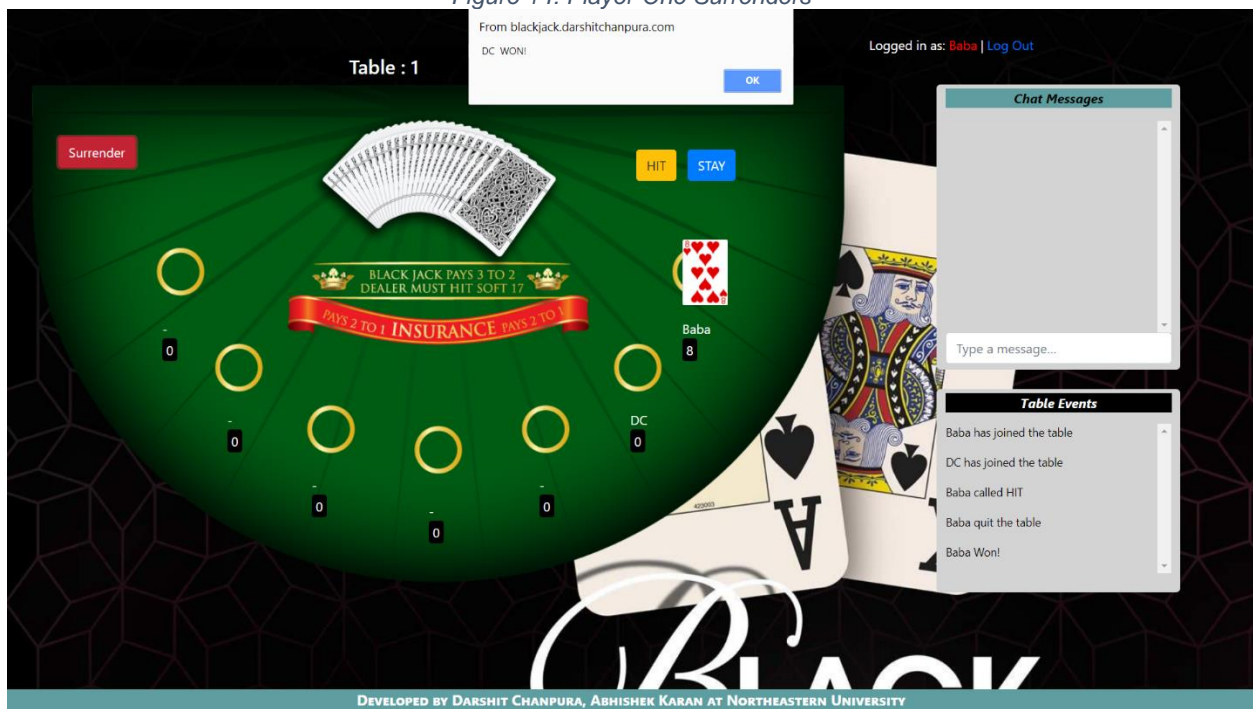


Figure 15: Player two wins

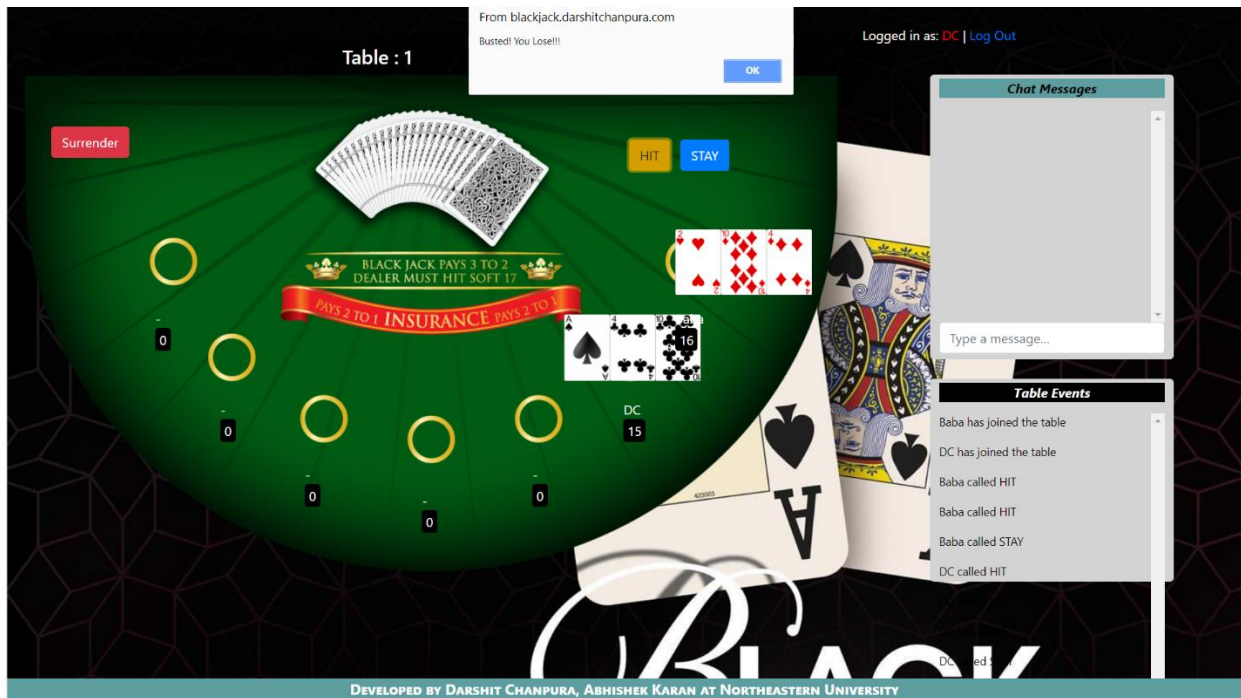


Figure 16: A player got Busted

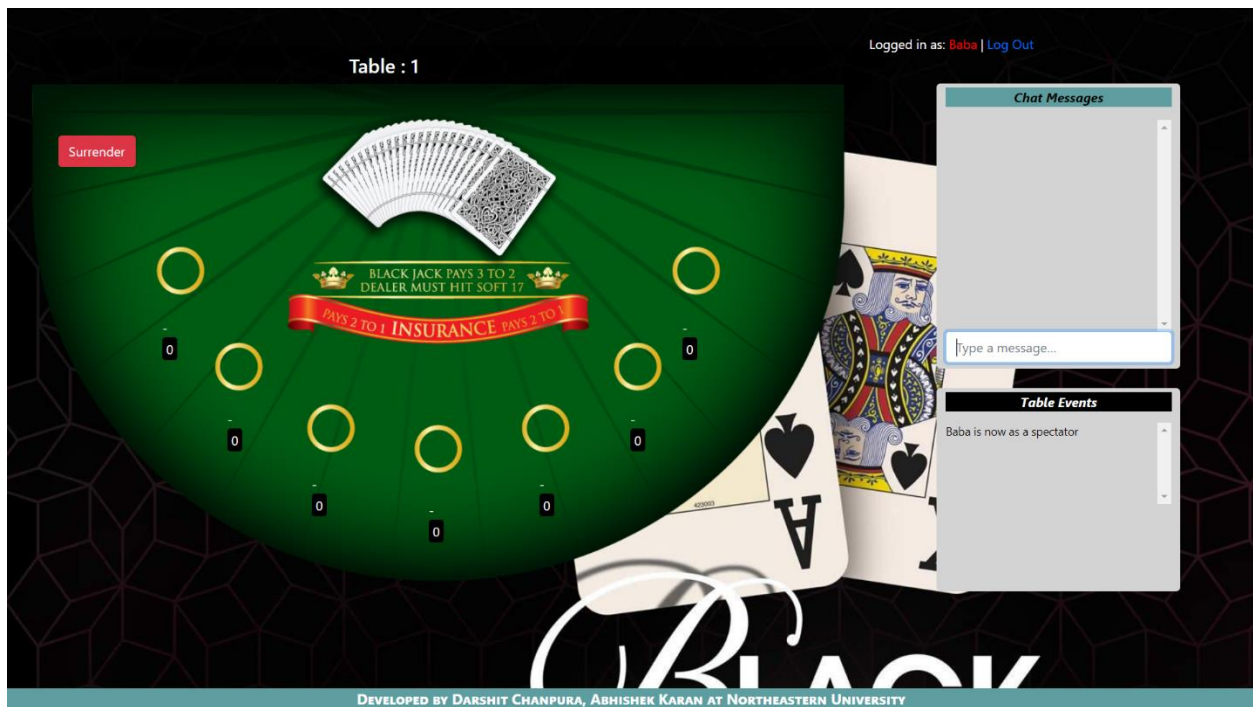


Figure 17: Spectator Joins

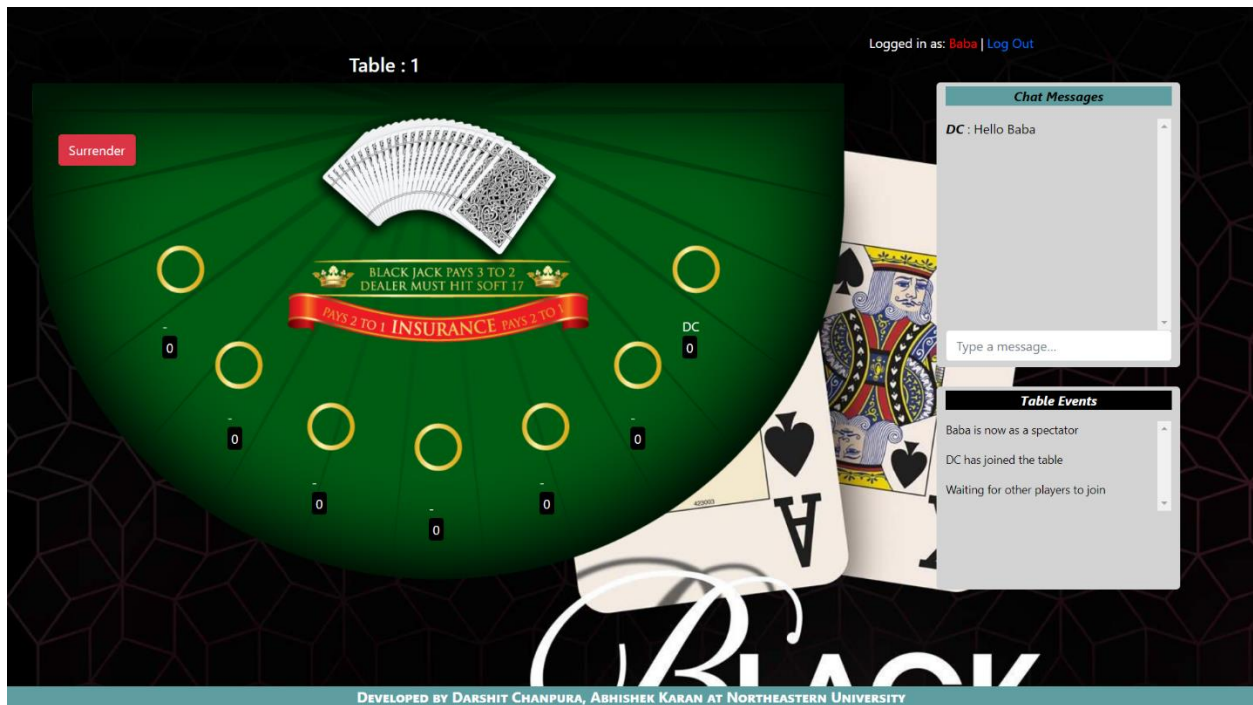


Figure 18: Player chatting with Spectator

References:

- [1] <https://en.wikipedia.org/wiki/Blackjack>
- [2] <https://www.pagat.com/banking/blackjack.html>
- [3] Card Images Attribution: Byron Knoll: <http://code.google.com/p/vector-playing-cards/>
- [4] Background Image: <http://www.merkur-online-casino.xyz/diz/online-casino.jpg>
- [5] Table background: <http://pharaohusa.com/wp-content/uploads/2017/09/blackjack-layout-4-green.png>
- [6] Chat: <https://medium.com/@Stephanbv/elixir-phoenix-build-a-simple-chat-room-7f20ee8e8f9c>
- [7] <https://hexdocs.pm/elixir/Agent.html>