Team TONS

Troy Carloni **O**livier Chan Sion Moy **N**athan Rennacker **S**ophia Chu

Final Project Presentation
Fall 2022

Game : UNO

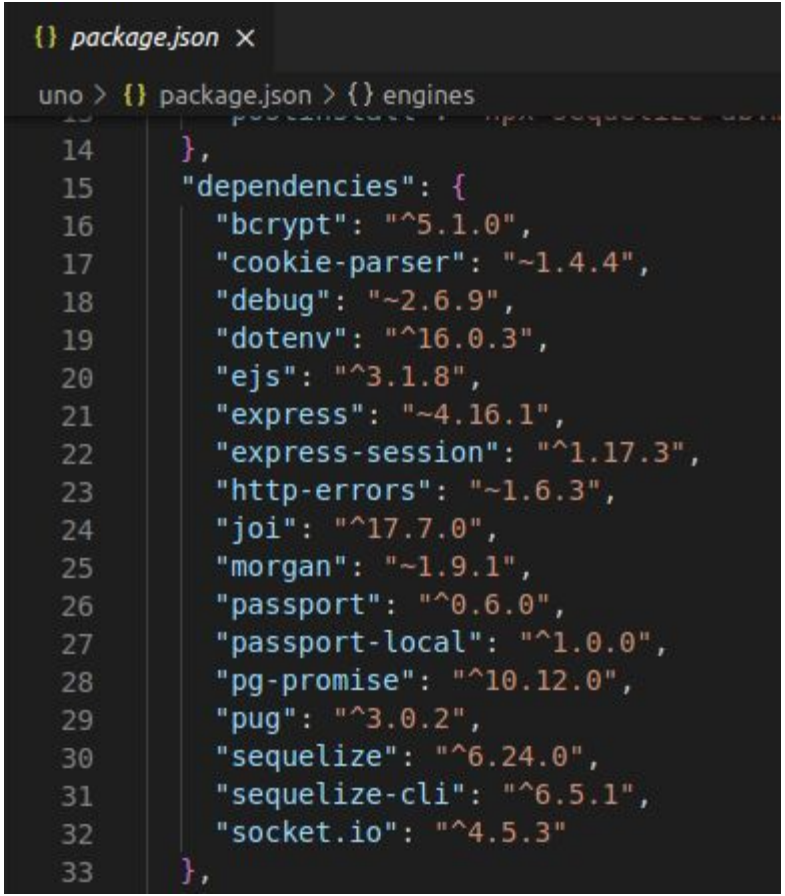

UNO RULES - QUICK START GUIDE
OBJECT
The first player to play all the cards in their hand wins.
SCORING & WINNING
If you are the first to get rid of all your cards, you get the points for the cards left in the other player's hands. Record the score after each hand.
SET UP
Choose the dealer.
Deal 7 cards each.
Place the reaming cards face down. This is the DRAW PILE.
Draw 1 card from the DRAW PILE and put it next to it.
This is the DISCARD PILE.
Player to the left of the dealer starts.
DRAW PILE
DISCARD PILE
LET'S PLAY!
Match the card on top of the DISCARD pile.
Color, number, and symbols are matches.
You can play the Wild Card and Wild Draw 4 card on any color.
If you can't play a card, draw 1 card from the DRAW PILE.
If it matches, you can play it at once.
You can only play 1 card at a time.
Before playing your second to the last card, you must say "UNO".
When a player plays their last card, the hand is over.
Points are tallied, and you start over.
UNO CARDS
Number Cards
Cards with Numbers 0 - 9
Allowed on: color/number
Scoring: Face Value
Draw 2 Cards
Next player draws 2 cards.
Allowed on: Color/symbol
Scoring: 20 points
Skip Cards
Next player looses turn
Allowed on: Color/symbol
Scoring: 20 points
Wild Cards
Changes color
Playable on all cards
Scoring: 50 points
Reverse Cards
Reverses direction of play.
Allowed on: Color/symbol
Scoring: 20 points
Wild Cards
Changes color
Next player draws 4
Playable on all cards
Scoring: 50 points
GOING OUT
If you forget to say "UNO" and the other players catch you - draw 4 cards.
If you forget and the player next to you begins his/her turn - you don't have to draw any cards.
Beginning a turn is defined as either drawing a card from the DRAW PILE or drawing a card from your hand to play.
If the DRAW PILE is depleted, reshuffle the DISCARD pile and continue play.
Special cards can be played as the last card.
If a Draw 2 or Wild Draw 4 card is played as the last hand, the next player should pick up the cards.
UNO RULES .ORG

Project setup

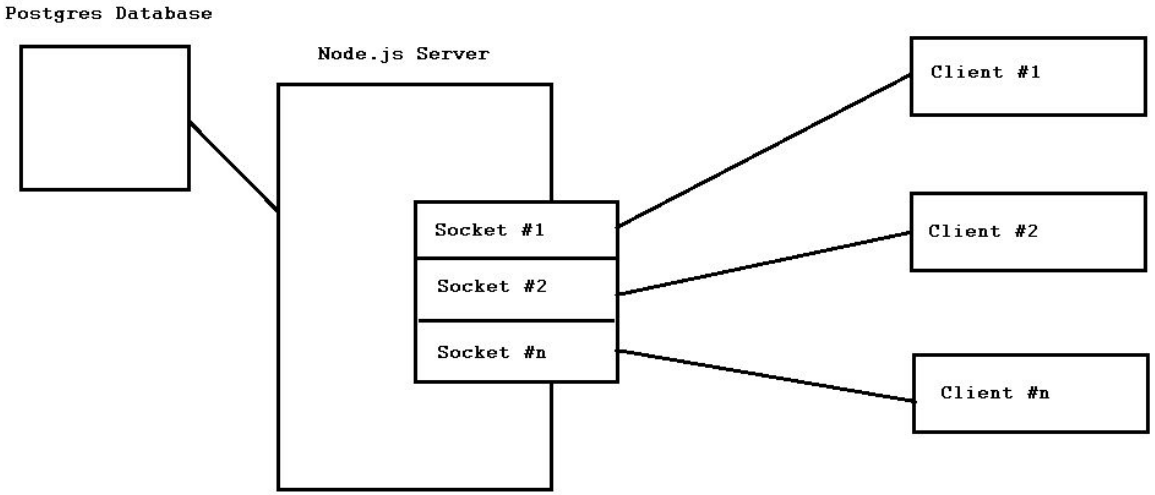
1. Create database in Postgres
2. Install dependencies and libraries
3. Define database schemas
4. Create routes to establish websocket connection
5. Implement logic for storing and retrieving data from db
6. Connect routes to frontend UI

Technologies

- Render
- Node.js
  - Express.js
  - Socket.IO
  - bcrypt (password hashing)
  - passport (Express.js authentication middleware)
  - joi (JSON schema validation)
- PostgresSQL



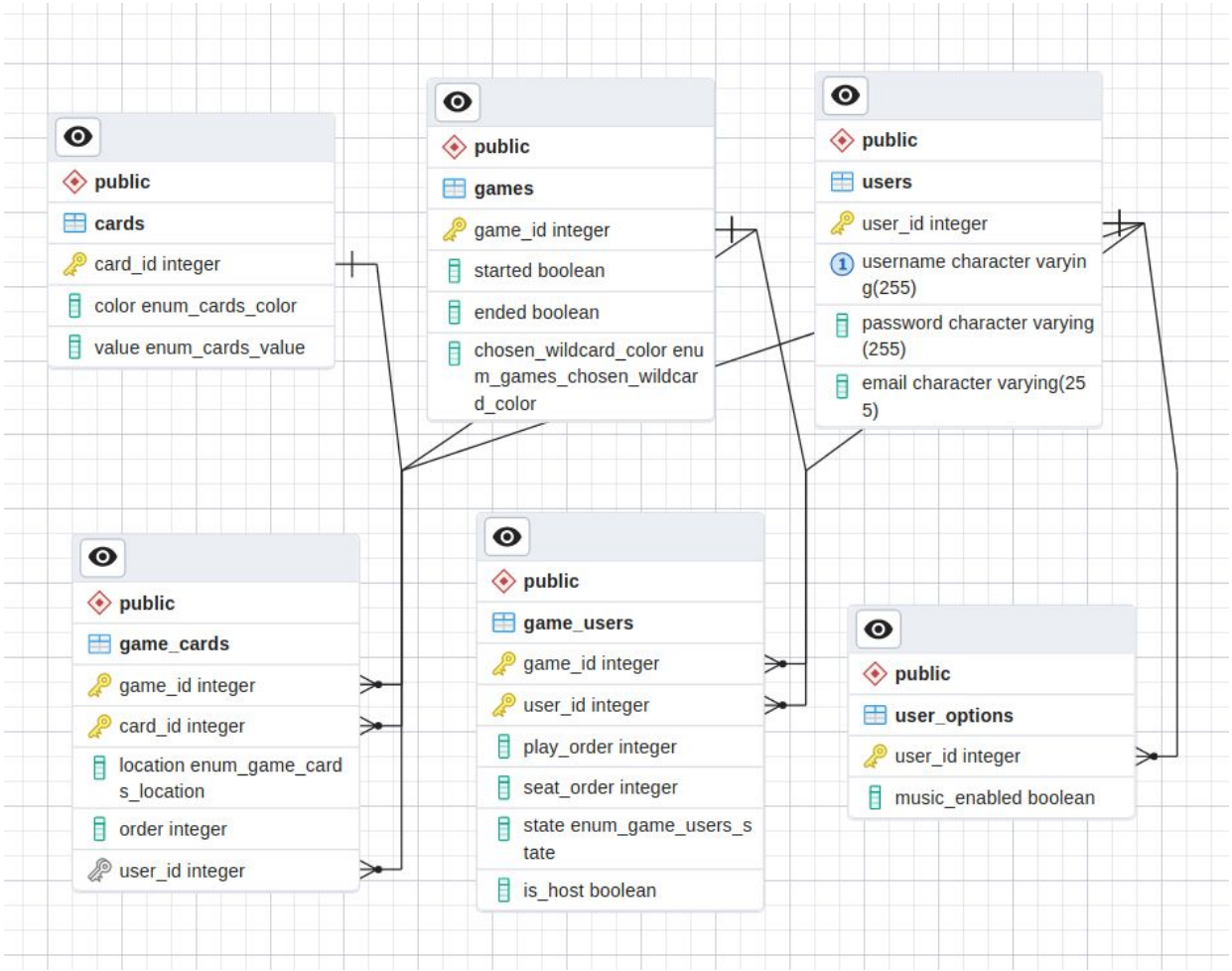
Architecture



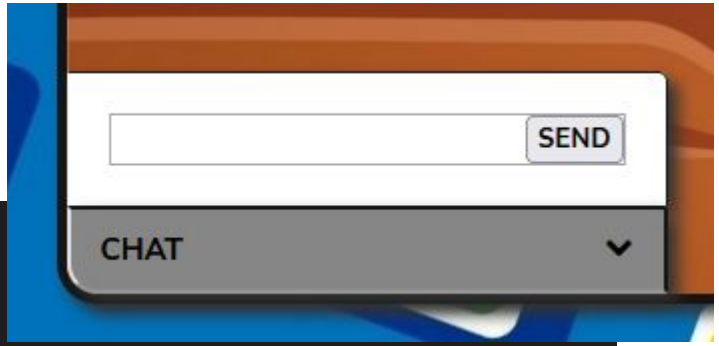
Sequelize Migration

Example migration shows:

- Creating new **table** called “**users**”
- Has following fields
  - ‘user_id’
  - ‘username’
  - ‘password’
  - ‘email’


public
cards
card_id integer
color enum_cards_color
value enum_cards_value
public
games
game_id integer
started boolean
ended boolean
chosen_wildcard_color enum_games_chosen_wildcard_color
public
users
user_id integer
username character varying(255)
password character varying(255)
email character varying(255)
public
game_cards
game_id integer
card_id integer
location enum_game_cards_location
order integer
user_id integer
public
game_users
game_id integer
user_id integer
play_order integer
seat_order integer
state enum_game_users_state
is_host boolean
public
user_options
user_id integer
music_enabled boolean

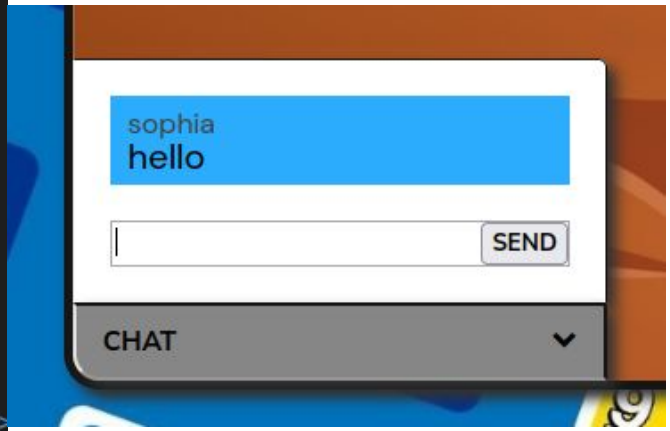
Global Chat example




```
15  const socket = io({
16    path: '/games/',
17    query: {
18      game_id: gameId,
19    },
20  });
21
22  // GCHAT functions
23  messageButton.addEventListener('click', addMessage);
24  socket.on('chat_message', (data) => {
25    message_container.innerHTML += createContainer(data.username, data.message);
26  });
```

User Sends Message

```
28  function addMessage() {
29    if (input.value === '') {
30      return;
31    } else {
32      var message = {
33        method: 'POST',
34        headers: {
35          Accept: 'application/json',
36          'Content-Type': 'application/json',
37        },
38        credentials: 'include',
39        body: JSON.stringify({
40          message: input.value,
41        }),
42      };
43      fetch(`/api/games/${gameId}/chat`, message).catch((err) =>
44        console.log(err)
45      );
46      input.value = '';
47      input.focus();
48    }
49  }
```



Problems we encountered

- Players were flying around the screen whenever a SKIP or REVERSE was played.
  - Players were displayed based on “play_order”. Our solution was to add another field called “seat_order” that never changed during the game.
- The Render instance was erroring every time we tried to register an account with “blob is not a constructor”.
  - Placed some files in the project root which Render looks for as a signal to use a different version of node.
- Cards in player’s hand would fly off the edges of the screen if there were too many cards in hand
  - More complex math to calculate the necessary overlap per card to stay within div

Test Plan

- Used Postman to test socket events before connecting frontend
- Liberal use of “console.log” to print received socket objects in browser
- Many play tests, both among ourselves using multiple browsers as well as 4-player games on Render website

Github Repository:
https://github.com/csc-667-fall-2022-sfsu-roberts/term-project-tons

Render Website
https://team-tons-uno.onrender.com