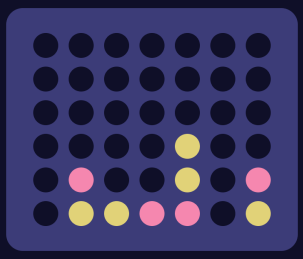
Maija Philip
Sept 26, 2024
ISTE 442

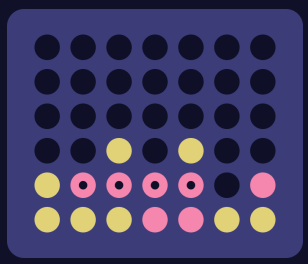
Connect 4 Proposal

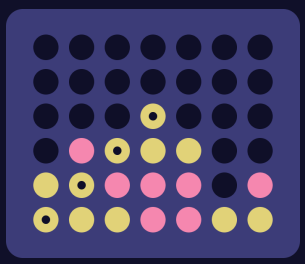
Gameplay

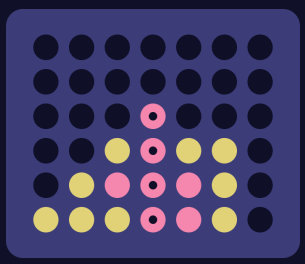


This is what the board of connect 4 looks like. It has slightly altered colors, but the same number of holes. How you play the game looks like this:

1. Person to start is randomized *(in this case, let's say pink goes first)*
2. Pink is able to drop a tile into any of the 7 columns. The tile will fall all the way to the bottom
3. Yellow is then able to drop a tile into any of the 7 columns. The tile will fall all the way to the bottom, but will not overlap another tile. *(there is one tile per space)*
4. Each player trades off dropping tiles into any of the columns. As multiple tiles are dropped into the same column, the tiles will stack in their respective spaces.
    a. Once a column's spaces are full, no more tiles can be placed in that column
5. In order for a player to win, they need to get 4 of their tiles in a row, column, or diagonal. Here is what some winning boards can look like:







6. If a player wants to end early, they can choose to forfeit the game and drop all the tiles from the bottom.

Technologies

React.js

Frontend

Node.js

Backend

MySQL

Database

Transport Mechanisms

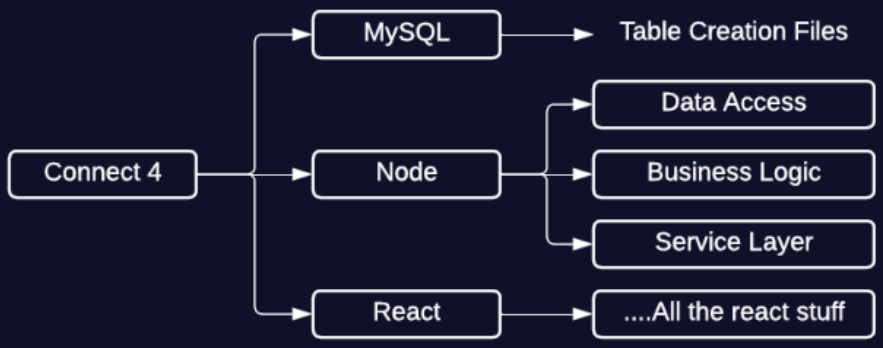
I will create a server which will be callable through certain APIs using Express.js in Node.js. From the frontend, I will use the Fetch API to call the APIs and fetch data similar to the way I did in this React App From Client Programming. The Fetch API allows me a simple way to make API calls from the frontend and has the ability for me to include data like headers and bodies.

For the chat functionality, I will use websockets to let the client know when there are knew messages and what those messages are. This is relatively simple in Node.js with the WebSockets library and Express.js.

Session Management Technologies

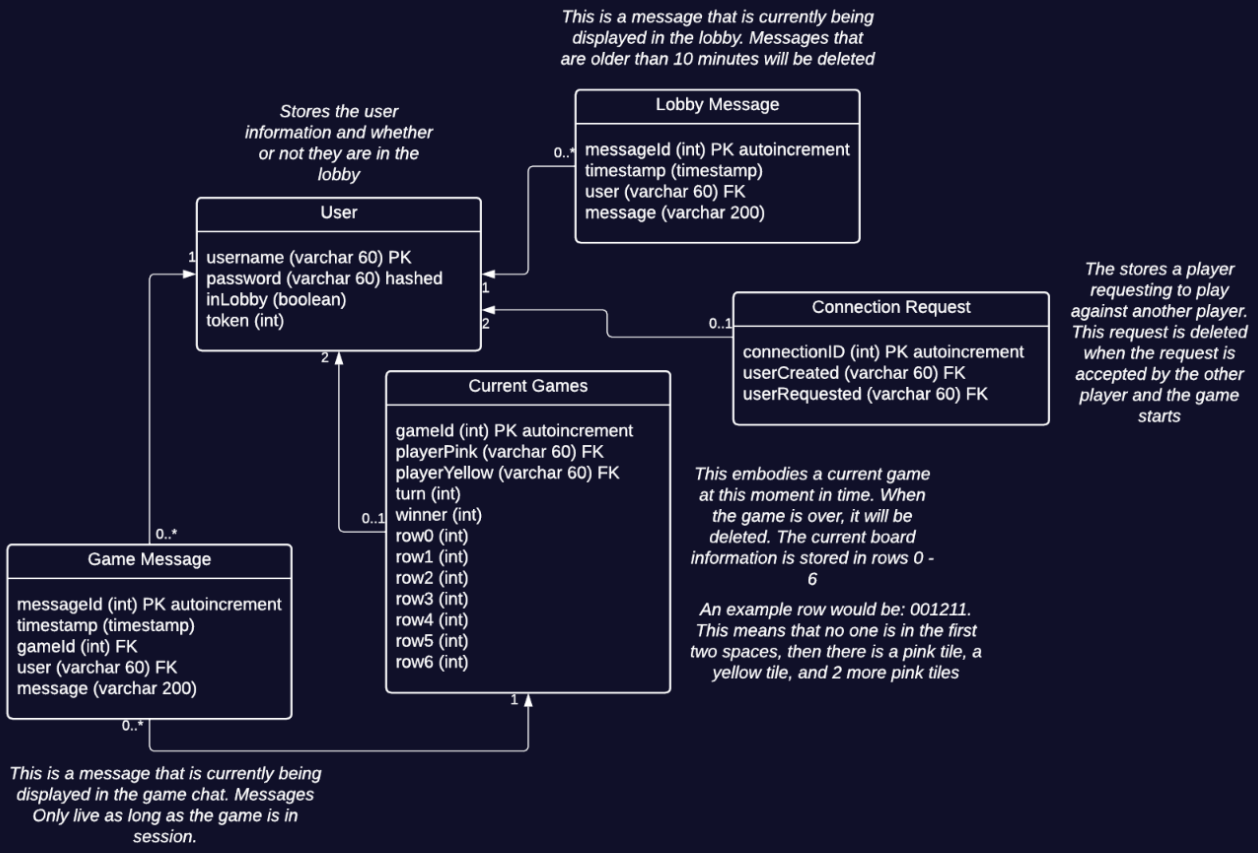
For session management, I will use an extension of Express, which is Express-Sessions which integrates session management into the api usage. In order to create secure tokens that the users can use within their session to make api calls, I will use the JSON Web Tokens library to help me create secure tokens.

Directories Diagram



ER Diagram

https://lucid.app/lucidchart/09802c0d-fa86-437a-bb43-3ae6eb3dc314/edit?viewport_loc=-55%2C-242%2C2366%2C1601%2C0_0&invitationId=inv_e55bc62e-eb98-4dd0-9a0a-cc14fde9792b



API Diagram

```
GET /user?username=”username”
      200 { username: “username”, inLobby: true }
      404 { error: “this user does not exist” }
      400 { error: “Please enter a username”}

      POST user/verfiyUser
            Content-Type: application/x-www-form-urlencoded
                  Payload: username, password
            200 { message: “logged in” }
            400 { message: “username or password is incorrect”}

      POST user/createNewUser
            Content-Type: application/x-www-form-urlencoded
                  Payload: username, password
            201 { message: “welcome” }
            400 { error: “please review the information and try again”}

GET /lobby
      200 [
            {
              messageId: 123456,
              timestamp: “2024-10-01 09:39:11”,
              user: “username”,
              message: “message”
            }
            … // returns all current lobby messages
          ]
      500 { error: “something went wrong, please try again” }

      POST /lobby/sendMessage
            Content-Type': 'application/json
                  { message: “message”, token: 123 }
            200 { message: “message sent” }
            404 { error: “token invalid” }
            500 { error: “something went wrong, please try again” }

      POST /lobby/sendGameRequest
            Content-Type': 'application/json
                  { username: “username”, token:  123 }
            200 { message: “request sent” }
```

```
            400 { error:“make sure both usernames exist and are in the
lobby”}
            500 { error: “something went wrong, please try again” }

GET /game/{gameId}
      200 {
            gameId: 123,
            playerPink: “username”,
            playerYellow: “username”,
            turn: 1 ,
            winner: 0,
            row0: 000000
            row1: 000000
            row2: 000000
            row3: 000000
            row4: 000000
            row5: 002000
            row6: 001210
          }
      404 { error: “game not found” }

      POST /game/{gameId}/getTurn
            Content-Type': 'application/json
                  { token:  123 }
            200 { yourTurn: false }
            404 { error: “token invalid” }

      POST /game/{gameId}/takeTurn
            Content-Type': 'application/json
                  { token:  123, xMove: 0, yMove: 6 }
            200 { message: “Turn taken” }
            400 { error: “make sure you have a valid move” }
            404 { error: “token invalid” }

      POST /game/{gameId}/forfeit
            Content-Type': 'application/json
                  { token:  123 }
            200 { message: “Game ended, your opponent won” }
            404 { error: “token invalid” }

      POST /game/{gameId}/sendMessage
            Content-Type': 'application/json
                  { message: “message”, token: 123 }
            200 { message: “message sent” }
```

404 { error: “token invalid” }
500 { error: “something went wrong, please try again” }