Final Project CPSC 2650

Tic Tac Toe

JULY 2022

GROUP MEMBERS:
Arminder Singh
Cesar Figueroa Socarras
Gustavo Carvalhaes de Sa

Overview

Tic-Tac-Toe is a final project for the CPSC 2650 class (Full Stack Web Development II).

The objective of this project is to apply concepts and practices seen in class to a real web application, enabling us to learn and actually implement resources and technologies.

Concepts implemented on this project:

1. User Authentication.
2. API development and testing.
3. Automation.
4. Containerization.

Project Features

The project consists of a multiplayer Tic-Tac-Toe game through the use of sockets.
An user can log to the application by using a google or github account. After a user is authenticated, he/she is able to create or join a game session/room and wait for another player to connect and therefore play, chat and interact.

Project features:

1. Authentication using social platforms (google and github accounts).
2. Create or join a game session/room.
3. Tic-Tac-Toe match against another user.
4. Chatbox to interact with your opponent.
5. Share game room/session ID through email.
6. Game scoreboard.

The project also contains an API for getting users and matches information (more information about the API can be found under the API documentation section).
Current endpoints implemented in the API:

1. GET all matches
2. GET match by id
3. POST match
4. GET user by id
5. POST user
6. POST email (used to share game id)

Tech Stack

The Tech Stack used in the project:

- React.js
- Node.js
- Firebase (authentication and database)
- Bootstrap

Automation

For the automation section we were able to combine Github Actions with Docker and Postman.

Every push or pull request to the main branch is thoroughly verified against each postman test case. If all the tests pass, the Github Workflow will proceed to build the docker image and finally accept the push/pull request to the main branch.

The repository is currently connected to Heroku, therefore every new commit that is accepted and merged/pushed to the main branch will be automatically deployed to Heroku.

Testing

For the project testing we used a postman collection for extensively verifying all the API endpoints responses.

Testing covered on the postman collection:

1. Response code status.
2. Posting endpoints with no data/wrong body format.
3. Data assertion in ID endpoints (ex.: GET user by id).
4. Error handling when request is unable to complete/not found.

Containerization

Docker was used for containerization of the project.
The docker image can be found under: docker image

API Documentation

The documentation for the API can be found on the project README