

Tic Tac Toe Game

Overview :- The game is created to play between two humans or human vs machine. The API is written in java (Spring) which uses H2 (in-memory database) for storing data and the frontend application is implemented using typescript (Angular-9) with hosting done on heroku

Note :- Only reason for using H2 as a database was it can be coupled into a java jar and no need for manual setup

Approaches / Algorithm

1. Winning algorithm

- a. **Vertical Condition**
Start from the topmost row of the game board to the bottommost excluding the row selected by the player for the column selected by him in the latest turn and check if all the values are equal to the player board value.
- b. **Horizontal Condition**
Start from the leftmost column of the game board to the rightmost column excluding the column selected by player for the row selected by him in the latest turn and check if all the values are equal to the player board value.
- c. **Diagonal 1 Condition**
Start from the first row, first column and move diagonally to the last row, last column excluding the row and column selected by the player in the latest turn and check if all the values are equal to the player board value.
- d. **Dianonal 2 Condition**
Start from the first row, last column and move diagonally to the last row, first column excluding the row and column selected by the player in the latest turn and check if all the values are equal to the player board value.
- e. If any of the above condition is true, then the player won the game

2. Computer game play algorithm

- a. **Priority 1**
Check if computer can win, fill that spot
- b. **Priority 2**
Check if human can win, fill that spot
- c. **Priority 3**
Check if computer can make two consecutive 'O' or 'X' with empty space, fill that spot
- d. **Priority 4**
Chose any empty spot available

References :-

1. <https://start.spring.io/>
2. <https://www.baeldung.com/spring-boot>
3. <https://angular.io/docs>
4. https://www.exploratorium.edu/brain_explorer/tictactoe.html

Deployment :- Deployment is also done on heroku and game can be accessed at

<https://rakshit-tic-tac-toe.herokuapp.com/>

The code for the assignment is present at :- <https://github.com/Rakshit2511/tic-tac-toe>

Base Documentation can be found at (You can check API from here as well just like postman) :-

<https://rakshit-tic-tac-toe.herokuapp.com/swagger-ui.html>

Health of the API can be checked at :- <https://rakshit-tic-tac-toe.herokuapp.com/actuator/health>

Base URL For API :- <https://rakshit-tic-tac-toe.herokuapp.com/tic-tac-toe/api/v1/>

Url	Method	action
/start	POST	startGame
/game-id/play	POST	playTurn
/	GET	getGames
/game-id	GET	getGame
/game-id/moves	GET	getMoves


Base URL For UI :- <https://rakshit-tic-tac-toe.herokuapp.com/>

Url	Method	action
/	GET	startGame
/play	GET	startGame

Steps to test the game :-

1. Go to above given url i.e :- <https://rakshit-tic-tac-toe.herokuapp.com/> and play the game

TIC TAC TOE GAME



Please fill the details to start the game

Human Vs Computer

First Player Name

Computer

O

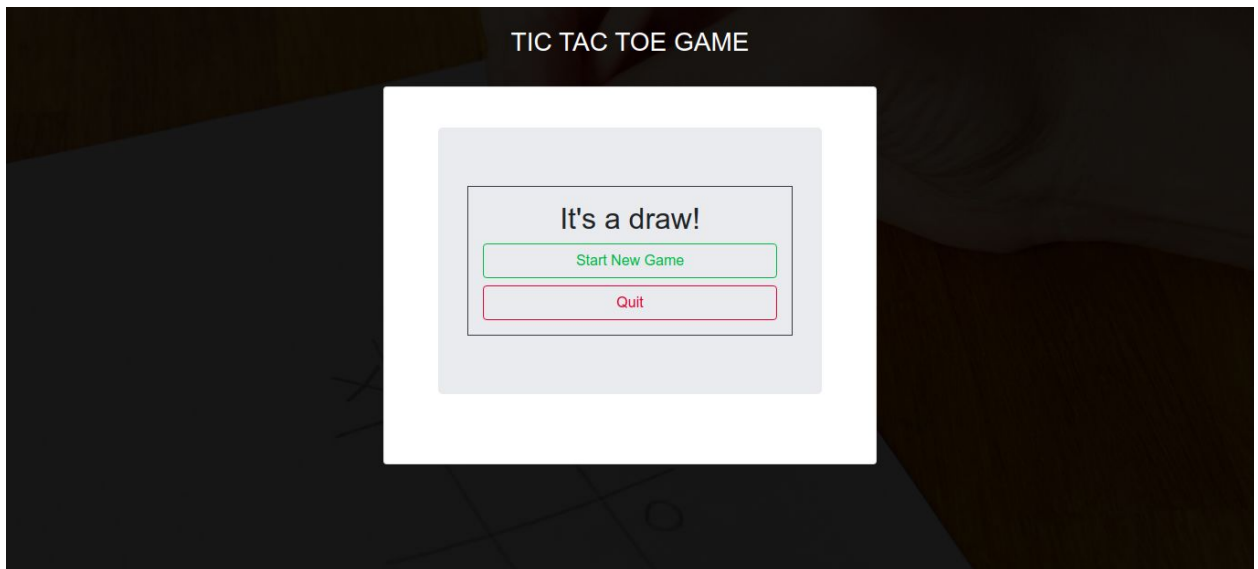
Matrix size

Start Game

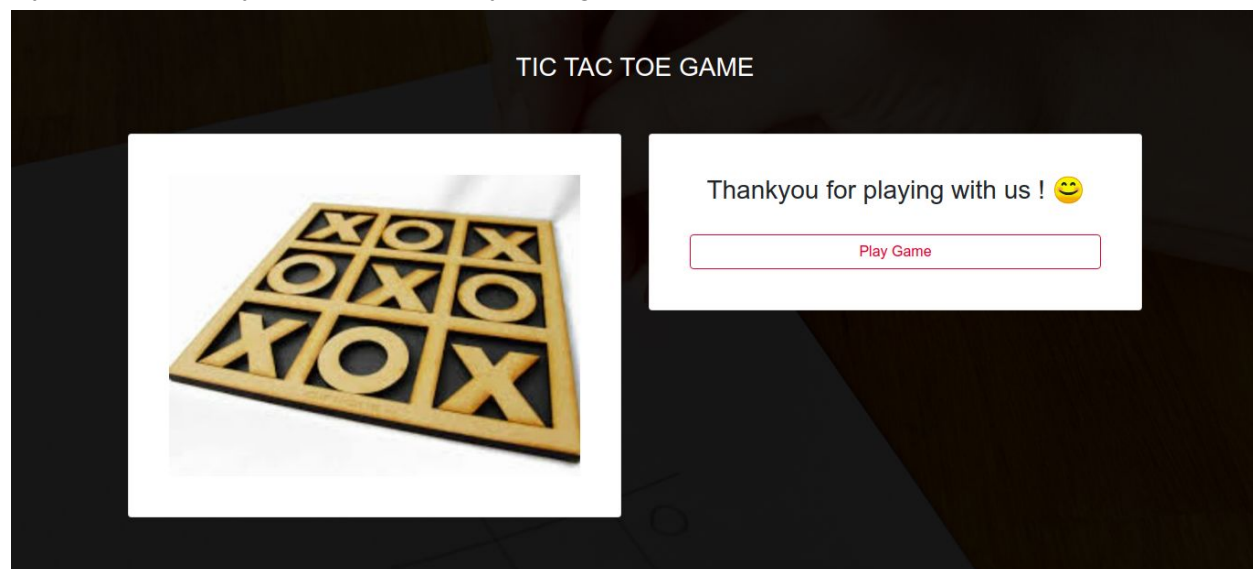
2. Select Human Vs Computer or Computer Vs Computer and enter the names :-
3. Select the board value of first player from 'O' or 'X' from the drop down
4. Select the matrix size for Example 3 for standard tic-tac-toe
5. Click on Start Game button
6. Start Playing the game

TIC TAC TOE GAME

Rakshit : O	Computer : X	
O	X	
	X	
		O



7. When games over, an alert is displayed :-
 - a. If you select Start New Game, you'll be back to home page
 - b. If you select Quit, you move to thank you page



8. Testing is complete

Steps to execute code and test solution on local :-

1. Clone the repository using the command :-
`git clone https://git-rba.hackerrank.com/git/4b70ff9e-7bec-4864-90eb-ce13e0d3376c`
2. Open to the backend code in intellij :-
`cd 4b70ff9e-7bec-4864-90eb-ce13e0d3376c/4b70ff9e-7bec-4864-90eb-ce13e0d3376c/tic-tac-toe`
3. Click on Add Configuration, then select maven and run it putting given command on command line :-
`spring-boot:run`

4. Visit localhost:8080/
5. No need to run the angular app since its packaged and added into spring-boot app
6. To run the java backend api using command line, package the code into a jar file and run using given command :-

```
java -jar -Dserver.port=8080 tic-tac-toe-0.0.1-SNAPSHOT.jar
```
7. Base Documentation can be found at :- <http://localhost:8000/swagger-ui.html>
8. Health of the API can be checked at :- <http://localhost:8000/actuator/health>
9. Database can be found at :- <http://localhost:8000/h2-console>
username :- rakshit
10. password :- secret
11. Base URL :- <http://localhost:8000/tic-tac-toe/api/v1/>

Steps to test the angular app separately (Not needed) :-

1. Clone the repository using the command :-

```
git clone https://git-rba.hackerrank.com/git/4b70ff9e-7bec-4864-90eb-ce13e0d3376c
```
2. Move to the folder

```
cd 4b70ff9e-7bec-4864-90eb-ce13e0d3376c/4b70ff9e-7bec-4864-90eb-ce13e0d3376c/tic-tac-toe-web-app/
```
3. Download required dependencies :-

```
npm install --save-dev @angular-devkit/build-angular
```
4. Run the angular app using the given command :-

```
ng serve --open
```
5. The web app will start at localhost:4200/
6. Don't forget to add

```
@CrossOrigin(origins = "http://localhost:4200")
```

above GameController in backend API before testing it like

```
@Slf4j
@RestController
@RequestMapping("/tic-tac-toe/api/v1")
@CrossOrigin(origins = "http://localhost:4200")
public class GameController {
```
7. We are good to go

Few things that are done in tic tac toe V1 :-

1. Basic documentation done and can be found at above given url
2. Basic validations done
3. Basic logging done
4. Basic UI creation done
5. Business logic done
6. Test cases created
7. Package creation done

Few Improvements that can be done in tic tac toe V2 :-

1. Add Spring security for session creation
2. Make the angular ui code more modular and clean
3. Use of Cloud config server for configuration at one end
4. Use of vault for saving db username and passwords
5. Use of onsite database to prevent data loss when api goes down
6. Add Eureka naming server and Ribbon for load balancing before deployment for handling large number of requests.