Dynamic Tic Tac Toe Game Using React.js

Project Report

Sushil Kumar Mahato

Date: July 10, 2025

1 Introduction

This project presents a responsive and dynamic version of the classic Tic Tac Toe game, developed using React.js. The application allows two players to engage in turn-based gameplay and supports customizable board sizes ranging from 3x3 to 10x10, enhancing the traditional gamefis flexibility. The primary objective was to deepen understanding of React concepts, including component-based architecture, state management with hooks (`useState`, `useEffect`), props, conditional rendering, and CSS styling.

2 Features

The Tic Tac Toe game includes the following features:

- **Dynamic Board Size**: Supports boards from 3x3 to 10x10.
- **Alternating Player Turns**: Players alternate between 'X' and 'O'.
- **Automatic Win Detection**: Detects winning conditions across rows, columns, and diagonals.
- **Draw Detection**: Identifies when the game ends in a draw.
- **Game Reset Button**: Allows restarting the game.
- **Responsive UI**: Styled using CSS and inline styles for a polished look.

3 Component Architecture

The application is structured into modular components for maintainability and scalability:

App.jsx Root component responsible for rendering the game and title.

Game.jsx Manages core logic, game state, win conditions, and dynamic board resizing.

Board.jsx Receives board data and renders the grid layout using CSS Grid.

Square.jsx Represents individual cells, styled as buttons with hover and active effects.

4 Technologies Used

The project leverages the following technologies:

- **React.js**: For building the user interface.

- **JavaScript (ES6+)**: For game logic and interactivity.
- **CSS**: For styling and responsive layout.
- **Vite (or React CLI)**: For project setup and build processes.

5 Screenshots

The following screenshots illustrate the gamefis interface:

- **Figure 1**: 3x3 Board with a Win (`./img/snip.png`).
- **Figure 2**: 5x5 Board (`./img/snip.png`).
- **Figure 3**: 4x4 Board (`./img/snip3.png`).

To view these images, open them from the `img/` folder.

6 Learnings

Through this project, the following skills were developed:

- Lifting and managing shared state in React.
- Efficient board rendering using array mapping.
- Implementing win and draw condition detection logic.
- Utilizing CSS Grid and Flexbox for responsive design.
- Structuring components for reusability and maintainability.

7 Challenges Faced

The development process presented several challenges, showcasing problem-solving skills:

- **Win Conditions for Larger Grids**: Adapting logic to handle dynamic board sizes.
- **Preventing Unnecessary Re-renders**: Optimizing React component updates.
- **Dynamic Board Creation**: Managing user input for variable board sizes.
- **Scalable Logic**: Ensuring clean and maintainable code for boards from 3x3 to 10x10.

8 Improvements and Next Steps

Future enhancements to the project include:

- **Single-Player Mode**: Implementing an AI opponent using the Minimax algorithm.
- **Theme Customizer**: Adding light and dark mode options.
- **Animations**: Enhancing cell updates with smooth transitions.
- **Keyboard Navigation**: Supporting gameplay via keyboard inputs.
- **Move History and Undo**: Allowing players to review and revert moves.