Tittle: Tic-Tac-Toe game

Table of contents

* **Project Overview**
* **Objective**
* **Technologies Used**
* **Key Features**
* **Game Logic**
* **Database Structure**
* **Challenges Faced**
* **Learnings**
* **Future Enhancements**
* **Snapshots**

# Project Overview

The project involves developing a real-time multiplayer Tic Tac Toe game using modern web technologies, ensuring efficient communication, smooth gameplay, and robust data handling.

# Objectives

* Create a seamless real-time multiplayer Tic Tac Toe experience with roombased matchmaking.
* Utilize modern web technologies such as ReactJS, TypeScript, Node.js, MySQL, and WebSocket API.
* Implement secure user authentication and session management.
* Ensure smooth gameplay through real-time synchronization, move validation, and game state updates.
* Design a database to store user information, game history, and results.
* Overcome challenges to improve application performance and reliability.
* Enable scalability for future enhancements like new game modes and ranking systems.

# Key Technologies

1. ReactJS: Used to build the dynamic and interactive user interface.
2. TypeScript: Improved code quality by adding static typing.
3. Node.js: Powered the backend for handling game logic and server requests.
4. MySQL: Managed data persistence, including user details and game records.
5. WebSocket API: Enabled real-time communication between players and the server.

**Features**

# User Interface

* Interactive UI for players to create or join rooms.
* Built with ReactJS for efficient rendering.

**Real-Time Multiplayer** - Players can:

* Create rooms with unique IDs.
* Join rooms by entering the room ID.
* WebSocket API ensures real-time communication and updates.

# User Authentication

* Players must register with unique usernames and passwords.
* A secure login system manages user sessions for authorized access.

# Game Logic

1. Game Board: Initialized as a 3x3 grid.
2. Move Validation: Ensures moves follow the game rules.
3. State Updates: Real-time synchronization of game state for both players.
4. Game End: Server determines the winner or if it's a draw and broadcasts results.

# Data Storage

* Game results stored in MySQL with the following schema:
* Player 1
* Player 2
* Winner

# Challenges

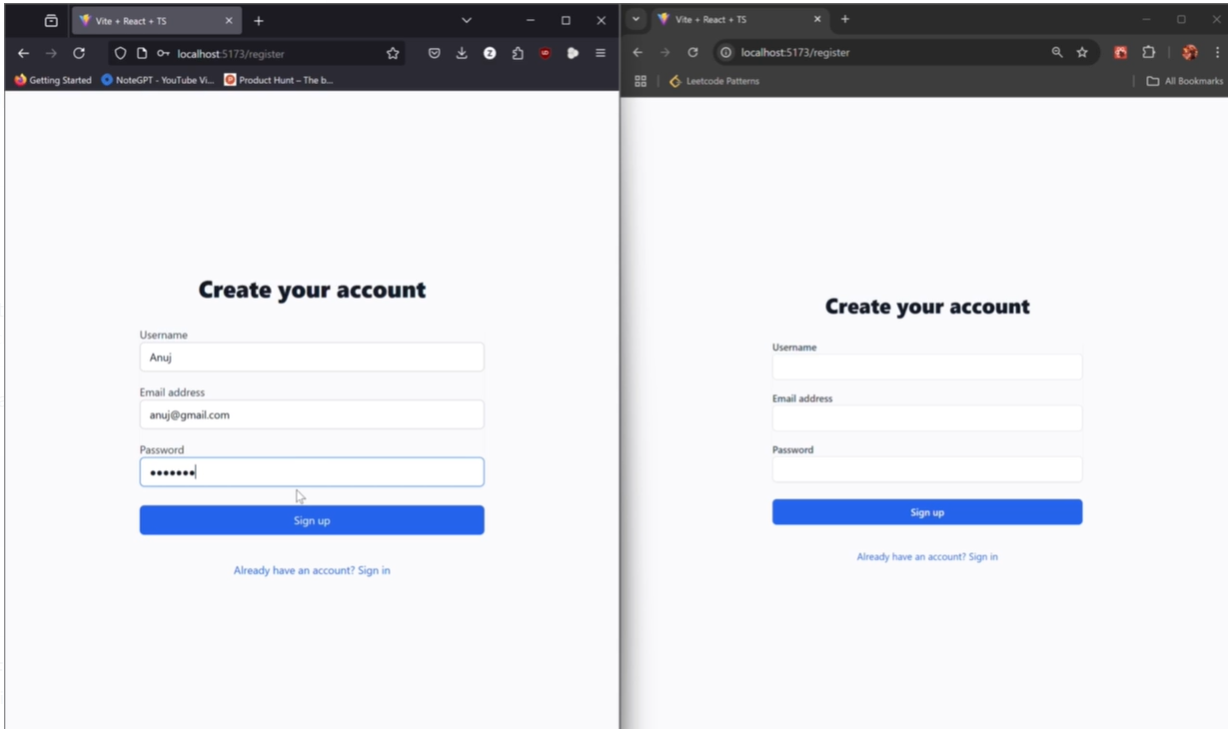
1. Managing multiple connections simultaneously.
2. Ensuring real-time synchronization between players.
3. Maintaining database consistency during high activity.

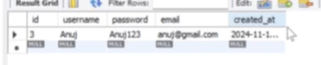
# Learnings

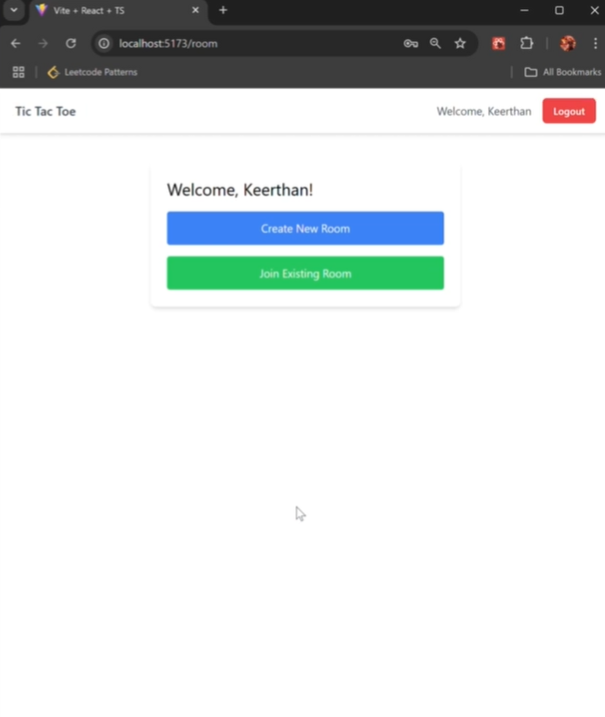
- Implementing real-time multiplayer games involves challenges such as handling concurrency, ensuring smooth gameplay, and maintaining data consistency.

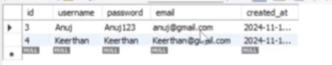
# Future Improvements

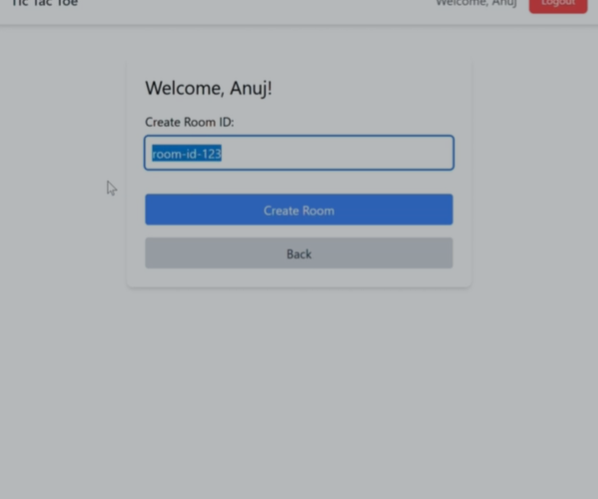
* Adding new game modes.
* Implementing a ranking system for players.
* Enhancing the user interface with modern design elements.

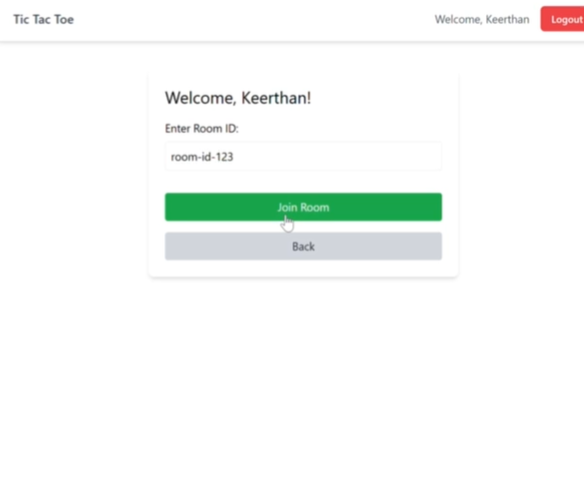


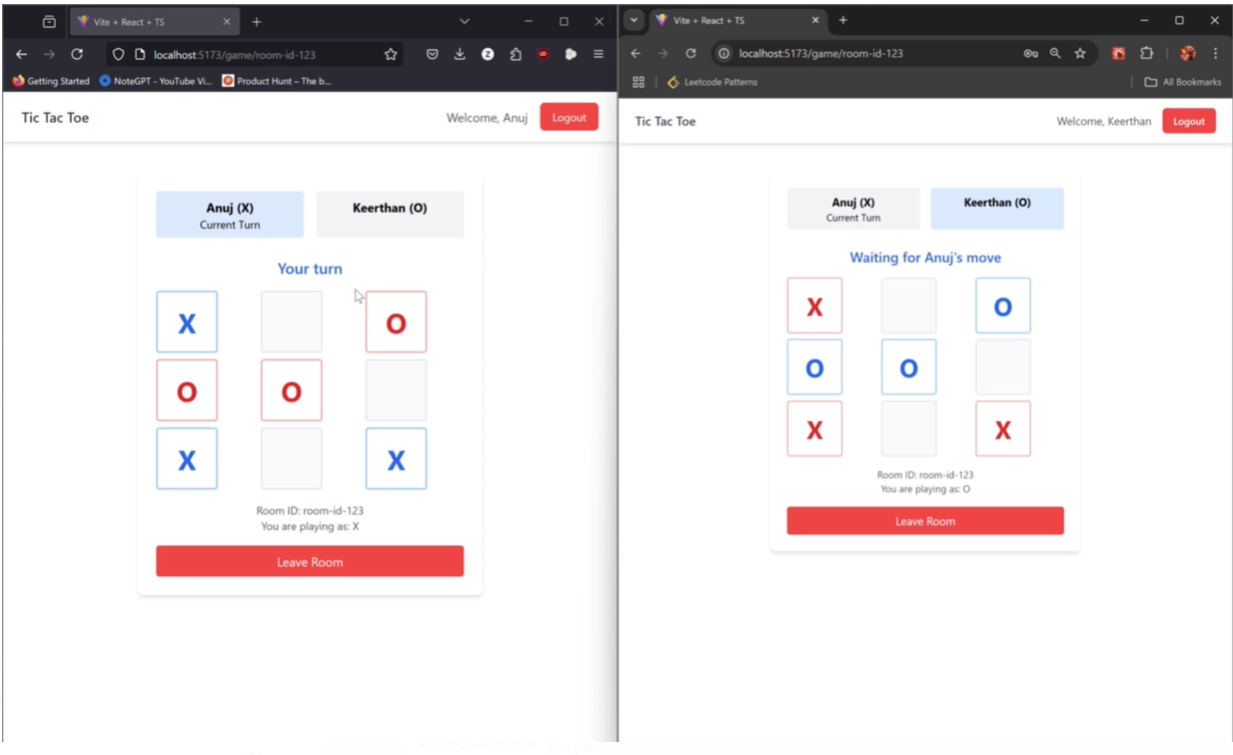


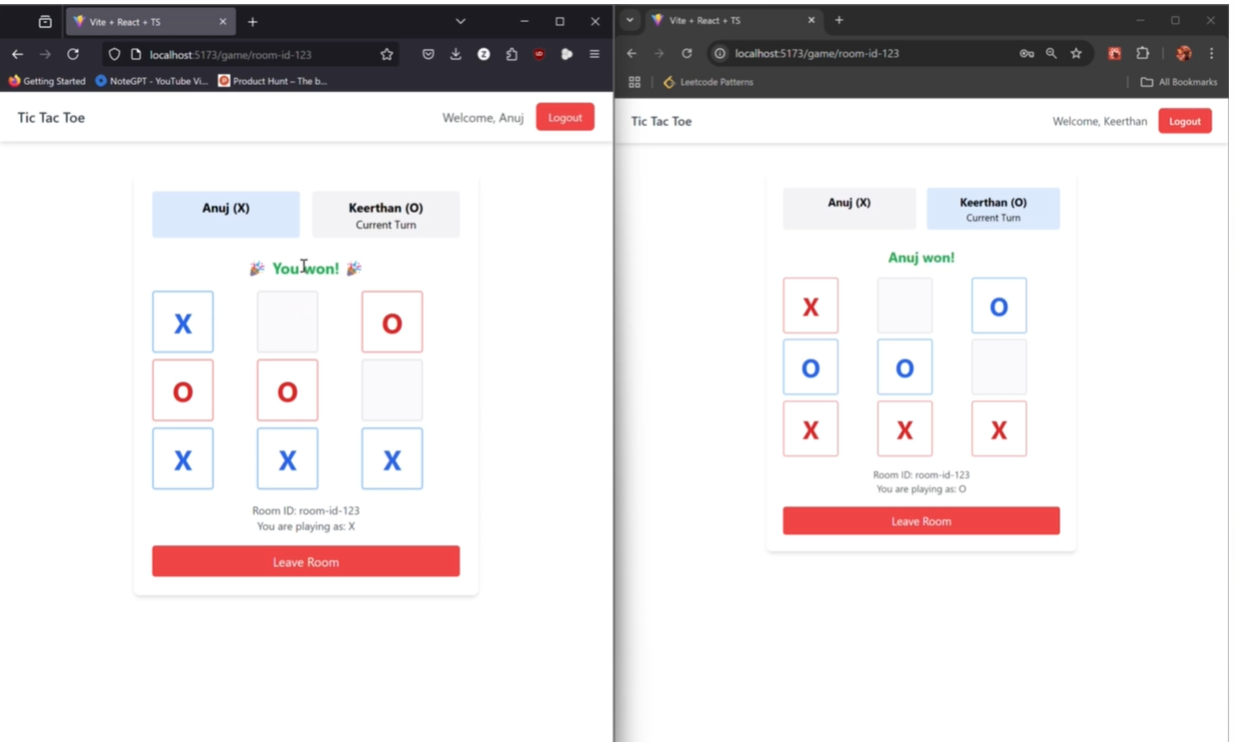












To run server and client in command prompt: cd server

npm run dev

cd client

npm run dev