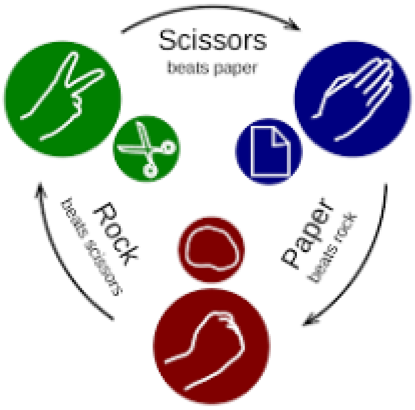
kontakt.io

Full Stack Engineer - Technical Assessment

Technical Assessment Prompt

Design and build a web app that allows you to play “Rock, Paper, Scissors” against a computer opponent and tracks their win/loss/tie totals. Use the standard version of the rules, where each turn the players must pick one of Rock, Paper, or Scissors,and the winner is determined by the below diagram.



UI

Web interface where players can start a game, enter their move for a round, and view their all-time statistics (win/loss/tie).

API

RESTful or RPC or GraphQL API and backend service(s) that is used by the UI to store state and play the game.

System Design

Database schema - written description of the database schema.
Architecture diagrams - visual description of the overall solution, detailing the entities and how data flows through the system.

Rock, Paper, Scissors Web Application Development

Summary

This report provides a comprehensive overview of the design and development process for a web application that allows users to play Rock, Paper, Scissors against a computerized opponent. The application is designed to be interactive, user-friendly, and capable of tracking detailed statistics of games played, including wins, losses, and ties. The development stack includes React with Vite and TypeScript for the frontend, Node.js with Express.js for the backend, and MongoDB for data persistence.

System Design and Architecture

Frontend Design

Technological Stack:

- **React:** Chosen for its efficient rendering and state management capabilities, which are essential for responsive UIs.
- **Vite:** A modern build tool that provides fast rebuilds and hot module replacement, enhancing developer productivity.
- **TypeScript:** Adds static typing to JavaScript, ensuring greater reliability and maintainability of the code.

User Interface Components:

- **Home Page:** This page features a simple layout with a prominent "Start Game" button that users can click to begin playing.
- **Game Interface:** Once the game starts, users can select their moves (Rock, Paper, or Scissors) through interactive buttons. The computer's move is generated randomly and both choices are displayed alongside the result of the round.
- **Statistics Dashboard:** This page allows users to view their historical performance, showing the number of wins, losses, and ties, as well as detailed logs of past games.

Backend Design

Technological Stack:

- **Node.js:** Provides a scalable environment for handling asynchronous operations and I/O bound tasks, which are common in web applications.
- **Express.js:** A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

API Endpoints:

POST /game/start: Initiates a new game session, resetting any session-specific data and preparing the game for new rounds.

POST /game/move: Handles the user's move, generates a computer move, evaluates the outcome, and updates ongoing statistics accordingly.

POST /game/end: Closes the current game session, archives its result, and can optionally start a new game.

GET /history: Retrieves the user's game history, including overall and detailed game statistics.

Database Schema

MongoDB Collections:

- **Games Collection:** Stores sessions with attributes such as session ID, user ID, start and end timestamps, and a summary of results (win/loss/tie counts).
- **Moves Collection:** Records each move in the game, detailing the user's choice, the computer's random choice, the result of the round, and timestamps.
- **Users Collection:** If user management is implemented, this would store user profiles, authentication data, and potentially preferences and settings.

Architecture Diagram Explanation

The architecture diagram visually represents the system's structure:

Implementation Considerations

Game Mechanics

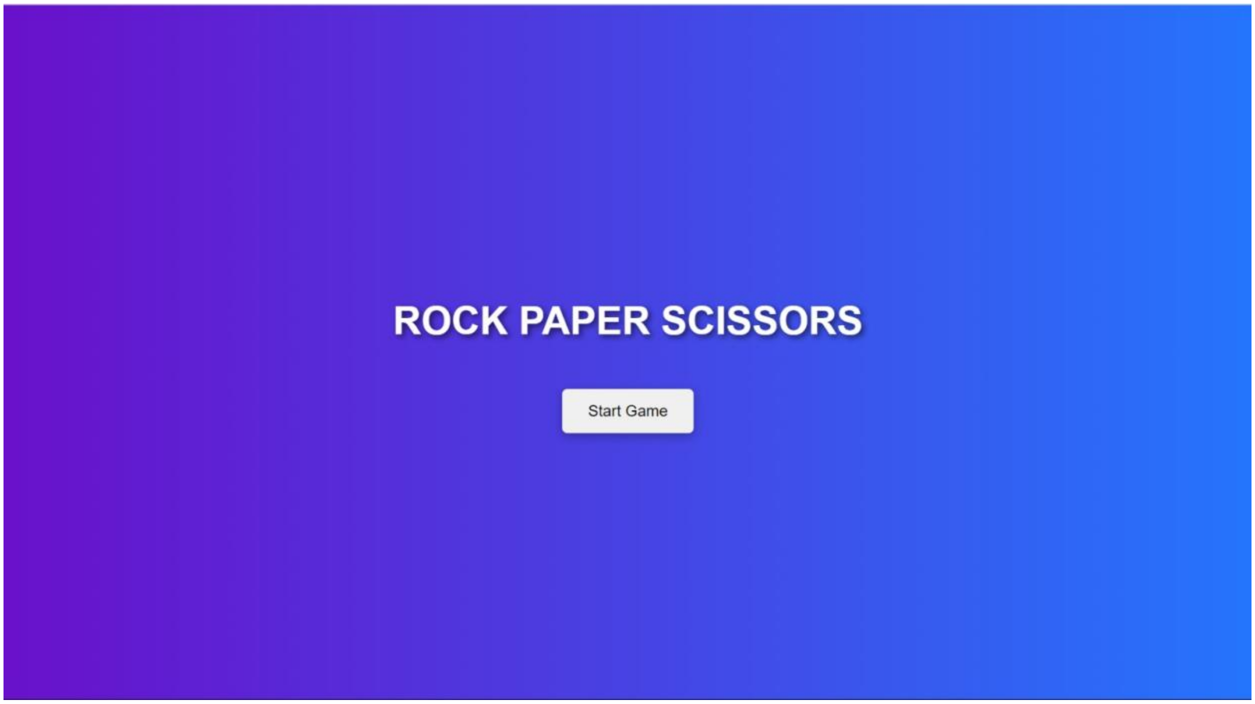
The application employs a simple algorithm to determine the outcome of each game round based on standard Rock, Paper, Scissors rules. This logic is handled server-side to prevent manipulation of the game results.

Security and Performance Enhancements

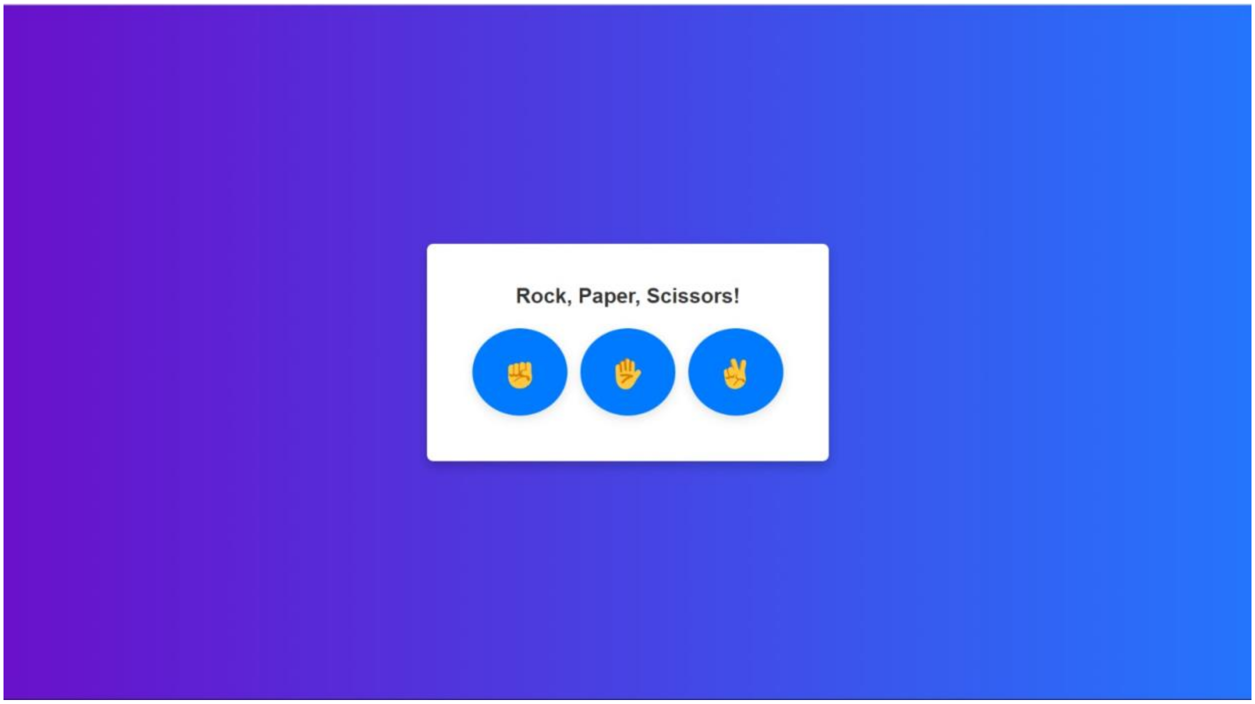
- **Input Validation:** Ensures that all user inputs are validated both client-side and server-side to prevent common web vulnerabilities.
- **Rate Limiting and Throttling:** Implemented to prevent abuse of the API, ensuring that the server remains available even under high load.
- **Indexing:** Essential for improving the performance of queries on the database, particularly for the history retrieval operations.

Conclusion

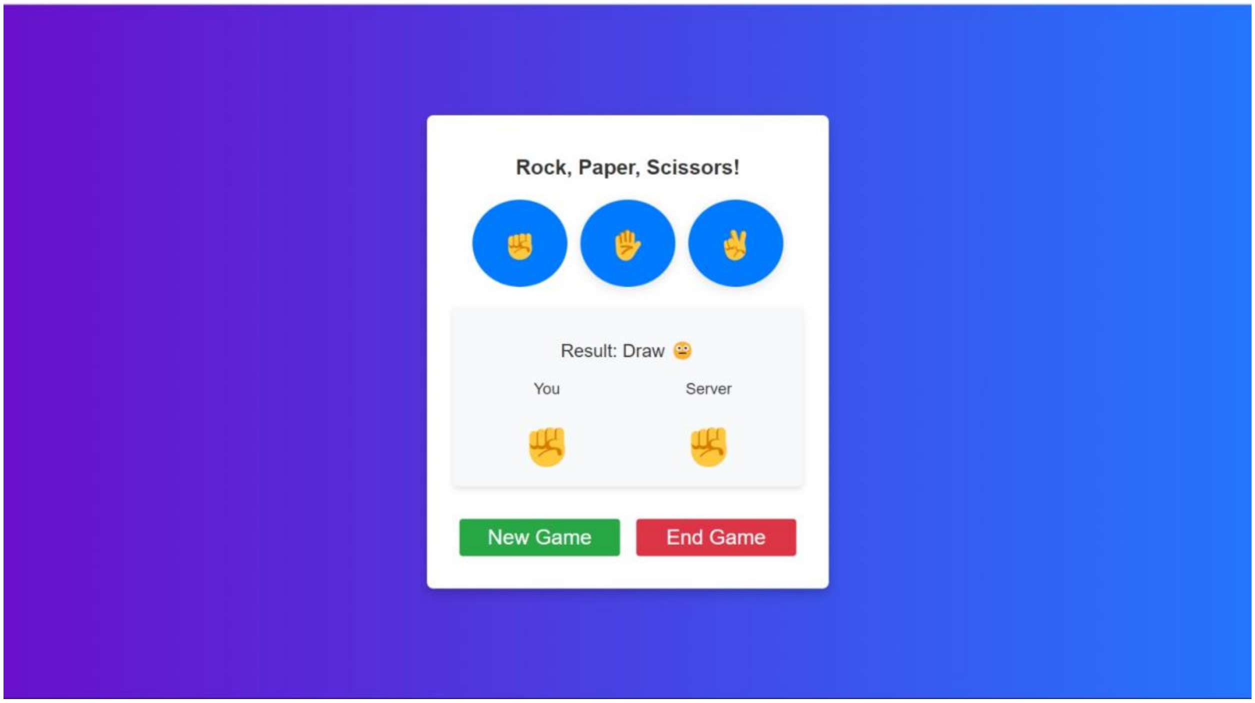
The Rock, Paper, Scissors game application uses modern web technologies to provide a robust and interactive user experience. Its architecture is designed to be scalable, maintainable, and secure. With the current setup, the application is well-prepared to handle a moderate to large number of users. Future enhancements could include real-time multiplayer capabilities, enhanced user profiles, and more comprehensive statistical tracking to enrich the gaming experience.



ROCK PAPER SCISSORS



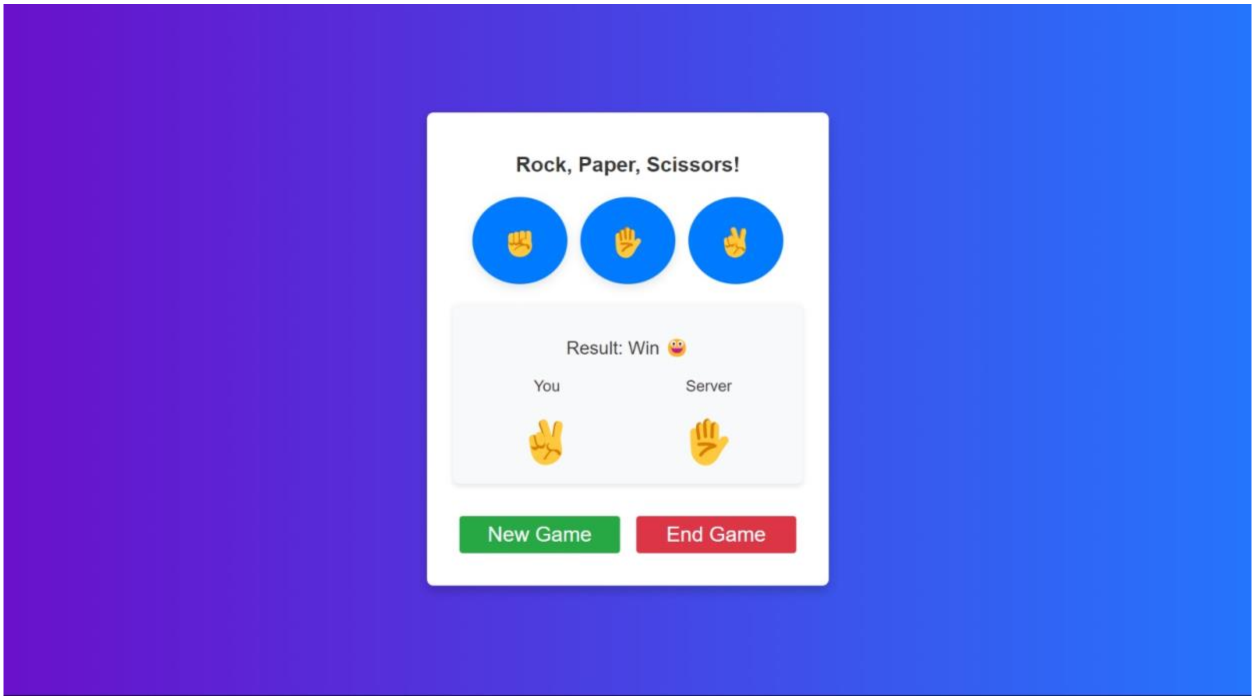
Rock, Paper, Scissors!



Result: Draw 😳

You ✊

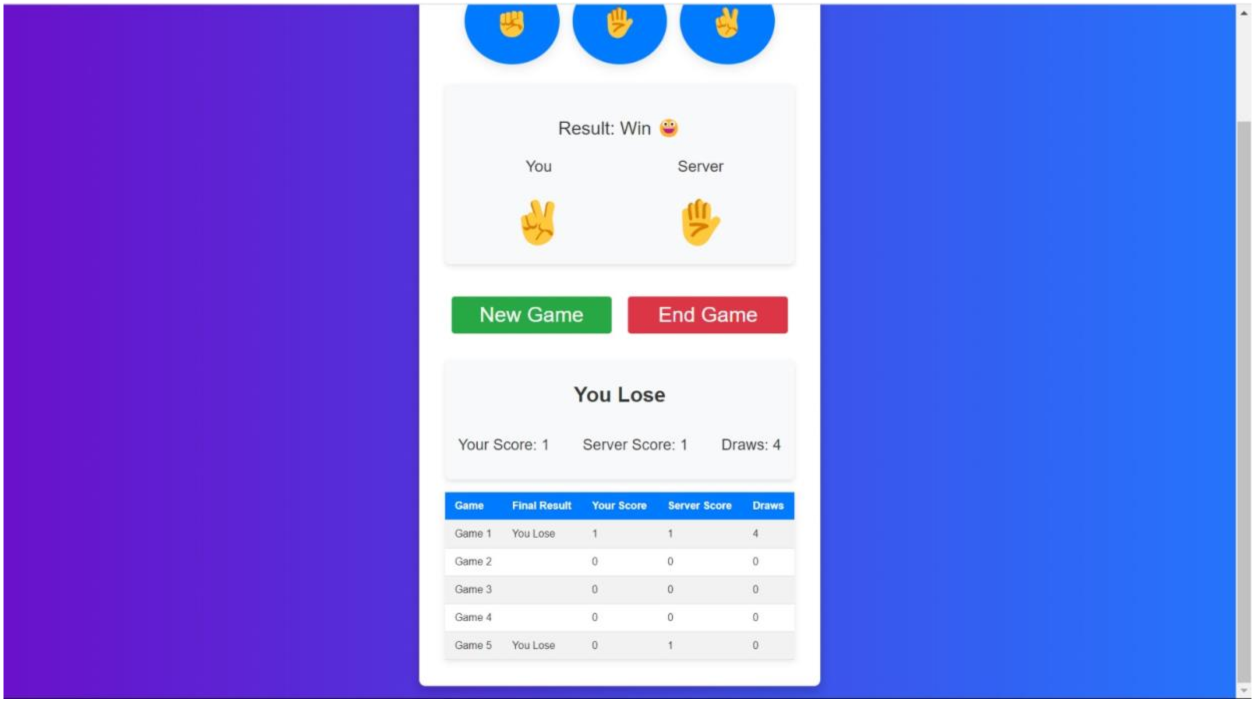
Server ✊



Result: Win 😃

You ✌

Server ✋



Result: Win 😜

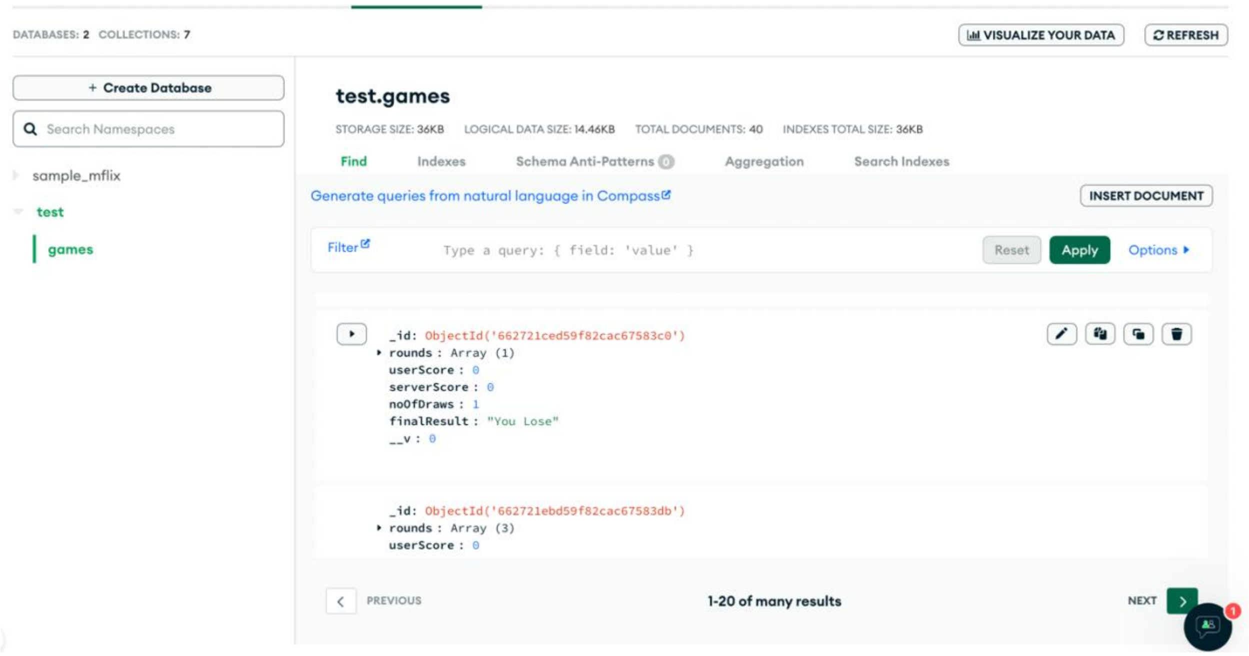
You Server

✌ ✋

You Lose

Your Score: 1 Server Score: 1 Draws: 4

Game	Final Result	Your Score	Server Score	Draws
Game 1	You Lose	1	1	4
Game 2		0	0	0
Game 3		0	0	0
Game 4		0	0	0
Game 5	You Lose	0	1	0



DATABASES: 2 COLLECTIONS: 7

test.games

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 14.46KB TOTAL DOCUMENTS: 40 INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('662721ced59f82cac67583c0')
▸ rounds : Array (1)
userScore : 0
serverScore : 0
noOfDraws : 1
finalResult : "You Lose"
__v : 0

_id: ObjectId('662721ebd59f82cac67583db')
▸ rounds : Array (3)
userScore : 0
```

1-20 of many results