Move37 Ventures Backend Developer Challenge: Real-Time Polling

Introduction

Welcome to the Move37 Ventures Backend Developer Challenge. This task is designed to assess your proficiency in building a robust backend service using modern technologies, with a specific focus on relational database design and real-time communication.

We are looking for a candidate who can demonstrate a strong understanding of **Node.js with Express, PostgreSQL, Prisma for ORM**, and **WebSocket implementation**. The project is a simplified version of a real-world polling application, allowing us to evaluate your ability to architect a solution that is both functional and scalable.

Project Goal: Real-Time Polling Application API

Your task is to create a backend service for a real-time polling application. The application needs to allow a user to create a poll, and other users to vote on it, with the results being updated instantly in real-time. The core functionality should be exposed via a RESTful API and a WebSocket layer for live updates.

Technologies Required

- **Backend Framework:** Node.js with Express.js
- **Database:** PostgreSQL
- **ORM:** Prisma
- **Real-time Communication:** WebSockets (using a library like ws or socket.io)

Core Requirements

1. **RESTful API:** Implement a set of API endpoints to perform CRUD operations on the following entities:
    - **User:** Create and retrieve users.
    - **Poll:** Create and retrieve polls, including their options.
    - **Vote:** Submit a vote for a specific poll option.
2. Database Schema & Relationships:
   This is the most critical part of the challenge. You must design and implement a PostgreSQL database schema using Prisma, correctly defining the relationships between the models.
    - **User Model:** id, name, email, passwordHash
    - **Poll Model:** id, question, isPublished, createdAt, updatedAt

- **PollOption Model:** id, text
- **Vote Model:** id

Your schema should correctly model the following relationships:

- **One-to-Many:** A User can create many Polls, but each Poll has only one creator (User). A Poll can have multiple PollOptions, but each PollOption belongs to only one Poll.
- **Many-to-Many:** A User can vote on many PollOptions, and a PollOption can be voted on by many Users. You must define this relationship correctly using a Prisma many-to-many relation, which will create a join table (e.g., Vote).

3. WebSocket Implementation:
   Implement a WebSocket server to handle real-time events. The server should broadcast updates to all connected clients for the following scenario:
   - **Live Results:** When a Vote is cast for a Poll, the updated vote counts for all PollOptions within that Poll should be broadcast to all clients who are currently viewing that specific poll.

Evaluation Criteria

Your submission will be evaluated based on the following:

- **Database Design:** Clarity and correctness of the Prisma schema, particularly how you've handled the one-to-many and many-to-many relationships.
- **Code Quality:** Clean, well-structured, and well-commented code. Adherence to best practices for a Node.js/Express application.
- **WebSocket Implementation:** Correct and efficient implementation of real-time communication for live poll results.
- **API Functionality:** The RESTful endpoints should be fully functional and handle data correctly.
- **Project Setup:** The project should be easy to set up and run with clear instructions (e.g., a README.md file).

Submission

Please submit a link to a public GitHub repository containing your complete project. The repository should include all necessary files to set up the project, including a package.json with dependencies, your Prisma schema, and a README.md file explaining how to run the application and test the endpoints.