Cricket Scoring App – First Round Technical Assessment

Overview

This assessment is designed to evaluate your practical skills in backend and frontend development, real-time communication, data modeling, and overall code structure. You are required to build a simplified version of a real-time cricket scoreboard using our tech stack.

Tech Stack

You are expected to use the following technologies:

- Backend: NestJS, TypeScript, MongoDB, Redis, Socket.IO
- Frontend: React with Next.js

Objective

Build a mini real-time cricket scoreboard that allows users to:

- Start a match
- Add ball-by-ball commentary
- View live match updates and commentary as they happen

Requirements

Backend (NestJS)

1. **Start a Match**
    - Endpoint: `POST /matches/start`
    - Accepts team and match information.
    - Generates a 4-digit unique match ID using an auto-increment counter (you have to also use _id another id would be user for different use case like unique key).
2. **Add Commentary**
    - Endpoint: `POST /matches/:id/commentary`
    - Adds commentary for a specific ball.
    - Fields should include: over, ball, event type (e.g., run, wicket, wide, etc.)

3. **Get Match Details**
    - Endpoint: `GET /matches/:id`
    - Returns match info with commentary.
4. **Database**
    - Use MongoDB to store matches and commentary data.
5. **Real-time Updates**
    - Use Redis and Socket.IO to broadcast commentary updates to connected clients.
6. **Auto-Incrementing Match ID**
    - Implement a reusable counter system that generates unique 4-digit match IDs.

Frontend (Next.js + React)

1. Display a list of ongoing matches.
2. Allow users to click on a match and view the full commentary.
3. Commentary should update in real time via WebSocket without page refresh.

Constraints

- Do not use any external UI libraries such as Tailwind CSS or Material UI.
- Do not use AI-generated code (e.g., GitHub Copilot, ChatGPT).
- Use plain CSS or inline styles only.
- Commentary updates must be delivered in real time using Socket.IO.
- Match IDs must be auto-incremented 4-digit numbers, not UUIDs.

Deliverables

- A GitHub repository with the following structure:

backend/
frontend/
README.md

* Include clear setup instructions to run both frontend and backend locally.

* Optionally, include a Postman collection or cURL examples for API testing.

Deadline

Submit the completed GitHub repository within 24 hours of receiving the assignment.

Evaluation Criteria

Category	Weight
Code structure and readability	20%
Backend implementation	20%
Frontend implementation	20%
Real-time communication	15%
MongoDB schema and logic	10%
Auto-increment match ID logic	10%
Documentation and setup guide	5%

Optional Bonus

- Implement pause/resume functionality for matches.
- Store and retrieve the latest 10 commentary entries from Redis cache.