Submission Instructions: Real-Time Engagement API

1. Code Repository Structure

The entire project must be submitted as a single Git repository (e.g., hosted publicly on GitHub, GitLab, or Bitbucket).

- **Repository Name:** awari-backend-assessment-api (or similar).
- **Root Folder Contents:** Must contain all configuration files necessary to run the project.

2. Required Files

File	Purpose	Requirement
README.md	**Primary Documentation**	**Mandatory** (See Section 3 below).
package.json	Dependencies	Must include all required dependencies (@nestjs/graphql, graphql-tools, graphql-subscriptions, graphql-redis-subscriptions, mongoose, etc.).
Dockerfile	Containerization	A robust file for building the application image.
docker-compose.yml	Local Setup	A file to spin up the entire stack: **NestJS API, MongoDB, and Redis** with a single command.

Source Code	src/ folder	Well-structured code reflecting a modular NestJS application (e.g., separate modules for Auth, Post, PubSub).

3. The README.md Guide (Crucial)

The README.md must function as a comprehensive guide for the reviewer. It should include the following sections:

3.1. Setup & Execution

- **Prerequisites:** List required software (Node.js version, Docker, Git).
- **Local Run Command:** Provide the single command to start all services locally (e.g., docker-compose up).
- **API Endpoint:** Specify the URL and port where the GraphQL playground or API will be available (e.g., http://localhost:3000/graphql).

3.2. Design Decisions

- **Data Model:** Clearly define the final **MongoDB Schema** for the Post document, explaining how you store and manage the likeCount, dislikeCount, and the list of users who have interacted with the post. **Justify** your choice of embedding vs. referencing for the like data.
- **Real-Time Flow:** Briefly describe the sequence of events when a user executes the likePost mutation (e.g., DB update $\rightarrow$ Redis PubSub publish $\rightarrow$ Subscription broadcast).
- **Security/Auth Mock:** Explain how the current user's ID is being mocked/retrieved for the purpose of the assessment.

3.3. Testing & Verification

- **Test Case 1 (Mutation):** Provide the full GraphQL mutation query (and variables) needed to **like a post**.
- **Test Case 2 (Subscription):** Provide the full GraphQL subscription query needed to **subscribe to updates** for that post ID.
- **Test Scenario Instructions:** Provide step-by-step instructions (e.g., "Open two browser tabs/tools, execute the Subscription in Tab 1, then execute the Mutation in Tab 2, and observe the instant update in Tab 1").

4. Submission Deadline

- The candidate must be clearly informed of the time limit (e.g., "The project should be completed and the repository link shared within **96 hours / 4 days** of receiving these instructions.").
- The email or message sharing the repository link should confirm that the solution meets all requirements and is runnable via the provided docker-compose.yml.

5. Interview Preparation

- The candidate should be informed that the follow-up technical interview will focus entirely on discussing the design decisions, code structure, and scaling considerations of this project.