Build a simple **Authentication, SocketIo** Real time notification **NestJS** with **JWT-based authentication, and Nest Web Socket Get way**.
The system should allow users to **register, log in, and access a protected profile route, and very simple p2p chatting**.

Endpoints

1. POST /auth/register

- **Input:** name, email, password
- **Action:**
  - Hash the password using **argon2**
  - Save the user in the database (**SQLite / Postgres**)
  - **Output:**
    { "message": "User created successfully" }

2. POST /auth/login

- **Input:** email, password
- **Action:**
  - Find the user in the database
  - Compare password with hashed password
  - If valid, generate a **JWT token**
  - **Output:**
    { "access_token": "JWT_TOKEN" }

---

3. GET /profile (Protected)

- **Access:** Requires **JWT token** in the request header (Authorization: Bearer <token>)
- **Action:**
  - Validate the JWT token using an **Auth Guard**
  - If valid, return user details
  - **Output:**
    { "id": 1, "name": "John Doe", "email": "john@example.com" }

---

4. GET way - chats, message (Protected)

- **Chats:** Must create with only 2 user id as reference
- **Message:** Message should contains (senderId, receiverId, content)

Technical Guidelines

- Framework: **NestJS**
- Password hashing: **argon2**

- Authentication: **JWT** (@nestjs/jwt, passport-jwt)
- Realtime chatting: **SocketIo**
- Validation: **class-validator** for DTOs
- Use an **Auth Guard** to protect the /profile route
- Database: **SQLite / Postgres / In-memory** (choose whichever is faster for you)
- Node-cache/Redis for In memory database

Deliverables

- Proper NestJS project structure (modules, controllers, services)
- DTOs for request validation
- Working JWT authentication flow
- Endpoints testable with **Postman** with proper documentations

Way to get more priority: (Bonus)

- Clean, readable and maintain able code
- Fully understandable API endpoint with tag, description and example
- Seed initial user testing from our side
- Creativeness, and Good practices in your code
- DRY (Don't Repeat Your Self) principle

Make sure you write your README.md file and also a deployed live link to test it.

NOTE: Don't try to use any AI (Artificial Intelligence), Otherwise you will get rejected