**NestJS Social Backend API Documentation**

**🚀 Project Overview**

This project is a **social backend API** built using **NestJS**, **PostgreSQL**, and **JWT-based authentication**. It provides functionalities for **user registration**, **login**, **friend management**, and **device settings** in a social networking environment.

**Features**

1. **User Registration**  
   Users can register by providing their **name**, **email**, and **password**. Optionally, users can also authenticate via **Google**.  
   Passwords are securely hashed before being saved to the database.
2. **User Login**  
   Authenticated users can log in with their **email** and **password**. The login generates a **JWT token**, which is used for subsequent authenticated requests.
3. **User Profile**  
   Users can access their profile information after login, which includes details like **name** and **bio**. Users can also update their profile data.
4. **Friend Requests**  
   Users can send friend requests to other users, which can be either **accepted** or **rejected**.  
   The system tracks the status of the request (pending, accepted, rejected).
5. **Friend Suggestions**  
   The API suggests **5 random users** excluding the current friends and self.
6. **List Friends**  
   Users can retrieve a list of all their **accepted friends**.

**💻 Technology Stack**

* **Backend Framework**: NestJS
* **Database**: PostgreSQL
* **Authentication**: JWT (JSON Web Tokens)
* **ORM**: TypeORM
* **Validation**: Class-Validator

**Setup Instructions**

**1. Install dependencies**

npm install

**2. Configure the environment file**

Create a .env file based on dev.env:

# PostgreSQL Database Settings

DB\_HOST=localhost

DB\_PORT=5432

DB\_USER=postgres

DB\_PASS=postgres

DB\_NAME=social\_app

# JWT Settings

JWT\_SECRET=your\_jwt\_secret

JWT\_EXPIRES\_IN=3600s

**3. Run the application**

npm run start:dev

The application should now be running at http://localhost:8000/api.

**API Endpoints**

**Authentication**

* **POST /api/auth/register**  
  Registers a new user.  
  Request Body:

{

"name": "John Doe",

"email": "john.doe@example.com",

"password": "password123"

}

* **POST /api/auth/login**  
  Logs in a user and returns a JWT token.  
  Request Body:

{

"email": "john.doe@example.com",

"password": "password123"

}

* **POST /api/auth/google-login** (Optional)  
  Google OAuth login integration (can be added later for Google auth).

**User Profile**

* **GET /api/users/profile**  
  Retrieves the authenticated user's profile.  
  **Authentication required** (Bearer token).
* **PATCH /api/users/profile**  
  Updates the authenticated user's profile (name, bio, etc.).  
  Request Body:

{

"name": "John Updated",

"bio": "Software developer."

}

* **GET /api/users**  
  Lists all users except the logged-in user.

**Friend Management**

* **POST /api/friends/request**  
  Sends a friend request to another user.  
  Request Body:

{

"userId": 2 // ID of the user to send a friend request to

}

* **PATCH /api/friends/request/:id/accept**  
  Accepts a friend request.  
  **Authentication required**.
* **PATCH /api/friends/request/:id/reject**  
  Rejects a friend request.  
  **Authentication required**.
* **GET /api/friends/list**  
  Lists all accepted friends for the authenticated user.  
  **Authentication required**.
* **GET /api/friends/requests**  
  Lists all incoming friend requests for the authenticated user.  
  **Authentication required**.

**Friend Suggestions**

* **GET /api/users/suggestions**  
  Provides 5 friend suggestions based on mutual connections or interests.  
  **Authentication required**.

**Security**

* **JWT Authentication**: The application uses JWT for securing endpoints.
  + The login endpoint (/api/auth/login) provides a JWT token.
  + The JWT token must be included in the Authorization header as Bearer token for any authenticated routes (e.g., GET /api/users/profile).

**Testing & Debugging**

To ensure the routes are working:

1. **Start the server**:  
   Make sure the server is running with npm run start:dev.
2. **Use Postman** or **curl** to test API routes.

Exampl:

curl -X POST http://localhost:8000/api/auth/register -H "Content-Type: application/json" -d '{"name":"saad","email":"saad@example.com","password":"123456"}'

**Final Notes**

This project implements a minimal **social network backend** with essential features like user registration, profile management, and friend requests. It uses **NestJS**, **PostgreSQL**, and **JWT** for authentication.