Node.js + TypeScript REST API - Endpoints Documentation

# Table of Contents

1. Installation  
2. Environment Setup  
3. API Endpoints  
 - User Endpoints  
 - Group Endpoints  
4. Testing  
5. Contributing  
6. License

# 1. Installation

1. Clone the repository:  
 ```bash  
 git clone https://github.com/your-username/your-repository-name.git  
 ```  
2. Navigate to the project directory:  
 ```bash  
 cd your-repository-name  
 ```  
3. Install the dependencies:  
 ```bash  
 npm install  
 ```  
4. Set up your `.env` file (if needed):  
 Create a `.env` file in the root directory of the project and add your database connection string, port, etc.  
 Example:  
 ```plaintext  
 DB\_HOST=localhost  
 DB\_USER=your-db-user  
 DB\_PASSWORD=your-db-password  
 DB\_NAME=your-db-name  
 PORT=3000  
 ```

# 2. Environment Setup

Ensure you have the following installed:  
- Node.js (v14 or higher)  
- TypeScript  
- PostgreSQL (or another database of your choice)  
To install TypeScript globally:  
```bash  
npm install -g typescript  
```  
To run the development server, use:  
```bash  
npm run dev  
```  
This will start the server on `http://localhost:3000`.

# 3. API Endpoints

## User Endpoints

### 1. Create User

POST /users  
Body:  
```json  
{  
 "name": "John",  
 "surname": "Doe",  
 "birth\_date": "1990-01-01",  
 "sex": "Male"  
}  
```  
Response:  
```json  
{  
 "id": 1,  
 "name": "John",  
 "surname": "Doe",  
 "birth\_date": "1990-01-01",  
 "sex": "Male",  
 "message": "User created successfully"  
}  
```

### 2. Get User by ID

GET /users/:id  
Response:  
```json  
{  
 "id": 1,  
 "name": "John",  
 "surname": "Doe",  
 "birth\_date": "1990-01-01",  
 "sex": "Male"  
}  
```

### 3. Update User

PUT /users/:id  
Body:  
```json  
{  
 "name": "John",  
 "surname": "Smith",  
 "birth\_date": "1990-01-01",  
 "sex": "Male"  
}  
```  
Response:  
```json  
{  
 "message": "User updated successfully"  
}  
```

### 4. Soft Delete User

DELETE /users/:id  
Response:  
```json  
{  
 "message": "User soft-deleted successfully"  
}  
```

### 5. Hard Delete User

DELETE /users/:id/hard  
Response:  
```json  
{  
 "message": "User deleted successfully"  
}  
```

## Group Endpoints

### 1. Create Group

POST /groups  
Body:  
```json  
{  
 "name": "Developers"  
}  
```  
Response:  
```json  
{  
 "message": "Group created successfully",  
 "group": {  
 "id": 1,  
 "name": "Developers"  
 }  
}  
```

### 2. Get All Groups

GET /groups  
Response:  
```json  
[  
 {  
 "id": 1,  
 "name": "Developers"  
 }  
]  
```

### 3. Get Group by ID

GET /groups/:id  
Response:  
```json  
{  
 "id": 1,  
 "name": "Developers"  
}  
```

### 4. Update Group

PUT /groups/:id  
Body:  
```json  
{  
 "name": "Advanced Developers"  
}  
```  
Response:  
```json  
{  
 "message": "Group updated successfully",  
 "group": {  
 "id": 1,  
 "name": "Advanced Developers"  
 }  
}  
```

### 5. Delete Group

DELETE /groups/:id  
Response:  
```json  
{  
 "message": "Group deleted successfully"  
}  
```

### 6. Join Group

POST /groups/:groupId/users/:userId/join  
Response:  
```json  
{  
 "message": "User successfully joined the group"  
}  
```

### 7. Leave Group

POST /groups/:groupId/users/:userId/leave  
Response:  
```json  
{  
 "message": "User successfully left the group"  
}  
```