Project Documentation

Project Overview

This project is a NestJS application with two main modules:

- 1- Authentication Module: Implements user registration, login, and role-based access control (RBAC) using JWT.
- 2- Product Management Module: Manages CRUD operations for products with admin-restricted actions.

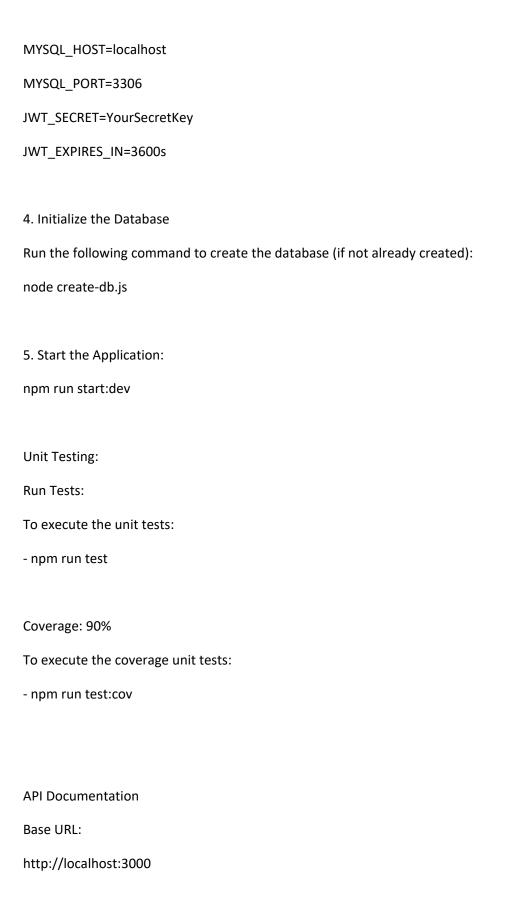
Technologies Used

- NestJS: Framework for building scalable Node.js applications.
- MySQL: Database for storing user and product data.
- TypeORM: ORM for database operations.
- @nestjs/jwt: Handles JWT-based authentication.
- Swagger: API documentation.
- Jest: Unit testing framework.

Features

- 1- Authentication Module:
- User registration with encrypted passwords.
- Login with JWT token generation.
- Role-based access control for routes.
- 2- Product Management Module:
- CRUD operations for products.
- Admin-only actions (add, update, delete products).
- Public access to view products.

3- Git Integration:
- Code is version-controlled using Git.
- Repository hosted on GitHub.
4-Unit Testing:
- AuthService and ProductService have comprehensive tests.
- Achieves 90% test coverage.
5- API Documentation:
Swagger provides detailed API documentation with examples.
Setup Instructions
1. Clone the Repository:
git clone <your-repository-url></your-repository-url>
cd <project-folder></project-folder>
2. Install Dependencies
Ensure you have Node.js and MySQL installed. Then, run:
npm install
3. Set Up the Environment
Add the following variables to app.module :
MYSQL_DATABASE=task
MYSQL_USER=root
MYSQL_PASSWORD=Password@12345



Authentication Endpoints:

```
1- Register a User
POST /user
Request Body:
  "email": "user@example.com",
  "password": "password123",
"role": "user"
}
Response:
{
  "statusCode": 201,
  "message": "User registered successfully"
"data": UserObj
}
2- Login a User
POST /user/login
Request Body:
{
  "email": "user@example.com",
  "password": "password123"
```

```
}
Response:
{
  "statusCode": 201,
  "message": "User registered successfully",
"access_token": "jwt-token",
"data": UserObj
}
Product Endpoints
1- Create a Product (Admin only)
POST /product
Headers:
{
  "Authorization": "Bearer <jwt-token>"
}
Request Body:
{
  "name": "Product 1",
  "description": "A great product",
  "price": 100,
  "stock": 50
}
```

2- List All Products

GET /product

3- Get a Product by ID

GET /product/:id

4-Update a Product (Admin only)

PATCH /product

5-Delete a Product (Admin only)

DELETE /product/:id