Overview

This document explains the process to set up and run a Node.js project that uses JWT (JSON Web Token) for user authentication. It covers project setup, installation of dependencies, running the server, and how to test the **Login**, and **Protected** APIs.

Project Setup

Prerequisites

Before starting, ensure that the following are installed on your system:

1. **Node.js and npm**: These are essential for running the application.
    - Download and install Node.js.
2. **MongoDB**: A MongoDB instance should be running, either locally or via a cloud service like MongoDB Atlas.

Step 1: Directory Structure

Create the following directory structure in your project:

```
product-api/
├── controllers/
│   └── product.controller.js
├── models/
│   └── product.model.js
├── routes/
│   ├── auth.routes.js
│   └── product.routes.js
├── middleware/
│   └── auth.middleware.js
├── swagger/
│   └── swagger.json
├── .env
├── server.js
├── package.json
└── README.md
```

models: Contains MongoDB schemas, such as the User schema.
routes: Defines API routes, including Register and Login.
middleware: Contains authentication middleware that checks JWT tokens.
server.js: The main file that sets up the Express server and connects to MongoDB.

Step 2: Install Required Dependencies

To install the necessary dependencies, navigate to your project directory and run:

npm install

This will install the following packages:

- **Express.js:** For creating the web server.
- **Mongoose:** For connecting to MongoDB and managing schemas.
- **jsonwebtoken (JWT):** For generating and verifying JWT tokens.
- **bcryptjs:** To securely hash and compare user passwords.
- **Body-parser:** To parse incoming request bodies.

Step 3: Setting Up MongoDB

You need a running MongoDB instance for user data storage. You can either:

- **Run MongoDB locally:** Make sure the MongoDB server is running with the command `mongod` in your terminal.
- **Use MongoDB Atlas:** Create a free cluster on MongoDB Atlas and get the connection string. Replace the connection string in your project configuration file.

Step 4: Configure the Server

- Create the server and connect it to MongoDB using the mongoose package.
- Set up middleware to handle request parsing and routing.

Running the Project

Step 1: Start MongoDB

Ensure that MongoDB is running locally or remotely. If using a local MongoDB instance, start it with:

Mongod

Step 2: Start the Node.js Server

Navigate to your project folder and run the following command to start the Node.js server:

node server.js

If you want automatic server restarts on file changes, use:

nodemon server.js

Once the server is running, you should see a message like:

Server running on port 3000

MongoDB connected

The server should now be listening on http://localhost:3000 by default

API Documentation

Swagger UI: http://localhost:3000/api-docs