Step-by-Step Guide to Building an API with Express and Mongoose

1. Project Setup

1.1 Initialize a Node.js Project

1. Open a terminal and create a new project folder.
2. Initialize a Node.js project.
3. Install necessary dependencies: Express, Mongoose, dotenv, CORS, bcrypt, JSON Web Token (JWT), and Pino for logging.
4. Install development dependencies like Nodemon for auto-restarting the server during development.
5. Create a structured project folder with subdirectories for models, routes, controllers, middlewares, and configurations.

2. Logger Configuration

2.1 Create a Logger Instance

1. Configure Pino to log to both console and a file.
2. Export the logger to be used across the application.

3. Setup Express Server

3.1 Configure Express and Middlewares

1. Import necessary modules and configure Express.
2. Apply middleware for JSON parsing, CORS handling, and logging.
3. Establish database connection.
4. Define a basic root route for testing API availability.
5. Start the server on a defined port.

4. Connect to MongoDB

4.1 Configure MongoDB Connection

1. Establish a connection to a MongoDB database using Mongoose.
2. Handle connection success and failure with logging.
3. Store database connection string in an environment variable.

5. Define the User Model

5.1 Create a User Schema

1. Define a User schema with fields: name, email, password, and role.
2. Hash passwords before saving them to the database.

6. Implement Authentication

6.1 Create Authentication Controller

1. Implement user registration.
2. Implement user login and generate JWT tokens.
3. Validate user credentials and return error messages for invalid attempts.

7. Define Product Model & Routes

7.1 Create Product Schema

1. Define a Product schema with fields: name, description, price, and stock quantity.
2. Store timestamps for product creation and updates.

7.2 Create Product Routes

1. Implement routes to create, read, update, and delete products.
2. Restrict access to modification and deletion based on user roles.
3. Allow public access to product listing and details.

8. Define Order Model & Routes

8.1 Create Order Schema

1. Define an Order schema with fields: user (reference to User), items (array of product objects), and total price.
2. Store timestamps for order creation and updates.

8.2 Create Order Routes

1. Implement order creation with authentication middleware.
2. Allow users to fetch their orders.
3. Restrict access to order data based on user authentication.

9. Implement Authentication & Authorization Middleware

9.1 Create Middleware for Auth

1. Verify JWT token from request headers.
2. Decode user ID and role from the token.
3. Restrict access to protected routes based on authentication.

10. Integrate Routes into the Server

1. Mount authentication routes.
2. Mount product and order-related routes with authentication where necessary.
3. Ensure middleware is correctly applied to protect sensitive routes.

11. Test & Deploy

1. Use Postman to test authentication, product, and order routes.

2. Deploy the API to a cloud platform such as Render or Vercel.
3. Set environment variables in the deployment environment.