1. Install Node.js on your machine. You can download it from the official website: https://nodejs.org/en/.
2. Install MongoDB on your machine. You can download it from the official website: https://www.mongodb.com/try/download/community.
3. Create a new project directory and navigate to it in your terminal.
4. Initialize a new Node.js project in the directory using the following command:
5. csharp
6. Copy code
7. npm init -y
8. Install the required dependencies by running the following commands in your terminal:
9. css
10. Copy code
11. npm install express mongoose cors body-parser
12. npm install --save-dev nodemon
13. Create a new file named server.js in your project directory and copy the server-side code from the above example into it.
14. Create a new file named db.js in your project directory and copy the database connection code from the above example into it.
15. Create a new file named user.js in your project directory and copy the user model code from the above example into it.
16. Create a new directory named routes in your project directory, and create a file named user.routes.js inside it. Copy the user routes code from the above example into it.
17. Create a new directory named controllers in your project directory, and create a file named user.controller.js inside it. Copy the user controller code from the above example into it.
18. Create a new directory named views in your project directory, and create a file named user-list.component.html inside it. Copy the HTML code from the above example into it.
19. Create a new directory named src in your project directory, and create a file named user.service.ts inside it. Copy the Angular service code from the above example into it.
20. Create a new directory named src/app in your project directory, and create a file named user-list.component.ts inside it. Copy the Angular component code from the above example into it.
21. Create a new directory named src/app/models in your project directory, and create a file named user.ts inside it. Copy the user model code from the above example into it.
22. Create a new directory named src/app/shared in your project directory, and create a file named environment.ts inside it. Copy the environment variable code from the above example into it.
23. Start your MongoDB server by running the following command in your terminal:
24. Copy code
25. mongod
26. Start your Node.js server by running the following command in your terminal:
27. Copy code
28. npm run dev
29. Navigate to http://localhost:4200/ in your web browser to view the application.
30. That's it! You should now be able to run the application locally and test the additional functionalities that we added.
31. npm install express mongoose cors body-parser: This command installs the necessary packages for your Node.js application. Here's what each package does:
32. express: A popular web framework for Node.js that provides a set of tools for building web applications.
33. mongoose: A library for MongoDB that makes it easier to work with MongoDB databases by providing a simple API for querying and manipulating data.
34. cors: A package that allows cross-origin resource sharing (CORS) in your application. CORS is a security feature that prevents web pages from making requests to a different domain than the one that served the web page.
35. body-parser: A middleware for parsing the request body in Node.js applications.
36. Running this command will install these packages and save them as dependencies in your project's package.json file.
37. npm install --save-dev nodemon: This command installs nodemon as a development dependency for your project. nodemon is a utility that will monitor for any changes in your source code and automatically restart your server. This is useful for development because it saves you from manually restarting your server every time you make a change. The --save-dev flag tells npm to save nodemon as a development dependency in your package.json file.