**Install Dependencies**

Npm install

**Configure MongoDB:**

Make sure you have a MongoDB server running.

Update the MongoDB connection string in **index.js** with your database information.

**Start the Application:**

Node index.js

node index.js

The application will be accessible at **http://localhost:3000**.

**Usage Instructions:**

**Register a User:**

Access **http://localhost:3000/auth/register** in your browser.

Fill out the registration form and submit.

**Login:**

Access **http://localhost:3000/auth/login** in your browser.

Enter your credentials and login.

**User Dashboard:**

After logging in, you'll be redirected to the user dashboard at **http://localhost:3000/tasks/dashboard**.

Here, you can view your tasks and perform various actions.

**Task Management:**

Create, update, prioritize, and delete tasks on the dashboard.

Use the search and filtering options to find specific tasks.

**Logout:**

Click on the logout link to log out of the application.

**Code Structure:**

**index.js:** Entry point of the application. Sets up Express, connects to MongoDB, and defines middleware.

**models/Task.js:** Defines the Mongoose model for tasks.

**middleware/auth.js:** Authentication middleware to protect routes requiring user authentication.

**routes/auth.js:** Handles user authentication routes (register, login, logout).

**routes/tasks.js:** Manages routes related to task management, including dashboard and task CRUD operations.

**views/:** Contains EJS views for different pages (dashboard, task list, create task, login).

**public/:** Stores static files like CSS, JavaScript, and images.

**middleware/:** Custom middleware functions.

**Additional Notes:**

**Environment Variables:**

If using sensitive information (like secret keys), consider using environment variables.

Update **.env.example** with your configurations and rename it to **.env**.

**Customization:**

Customize views, styles, and scripts in the **views/** and **public/** directories.

**Security Considerations:**

Implement security best practices, especially if deploying to production.

**Dependencies:**

**express**: Web framework for Node.js.

**mongoose**: MongoDB object modeling for Node.js.

**ejs**: Embedded JavaScript templates.

**express-session** and **cookie-parser**: Middleware for managing sessions and cookies.

**Development and Contribution:**

**Contributing:**

Fork the repository, create a branch, make changes, and submit a pull request.

**Testing:**

Thoroughly test the application, especially authentication and task management functionalities.

**Issues:**

Report any issues or bugs on the GitHub repository.

Remember to continuously update your documentation as your project evolves. Providing clear and comprehensive documentation ensures that others (or future you) can understand, use, and contribute to your project effectively.