|  |
| --- |
| ***Task Manager Web Application*** |

***Team Leads (Software Development Internee)***

***Prepared by:***

***Aliza Waseem***

***Intern-Software Development***

Words of Gratitude

***This internship has been a truly rewarding experience, and I am sincerely thankful to everyone who played a part in it.***

***I would like to extend my deep appreciation to Mr. Faizan Khan for his patient mentorship, constructive feedback, and steady encouragement that guided me throughout the project.***

***My heartfelt thanks also go to the development team at Team Leads for fostering an environment where learning and collaboration were always welcomed.***

***I’m especially grateful to my family and friends whose constant motivation gave me the confidence to stay dedicated and focused from start to finish***.

11. Appendix

 11.1 GitHub Repository Link

 11.2 Demo Video Link

 11.3 Contact Information

***1. Introduction***

***1.1 Internship Context & Background***

***1.2 Purpose and Learning Objectives***

***1.3 Overview of the Organization***

***2. Technology Overview***

***2.1 Tools Used for Frontend Development***

***2.2 Server-side Technologies (Backend)***

***2.3 Database Systems & Supporting Tools***

***3. Application Summary***

***3.1 Title of the Project***

***3.2 Identified Problem & Scope***

***3.3 Project Goals and Approach***

***3.4 Key Functionalities Delivered***

***4. System Design & Architecture***

***4.1 Project Directory Layout***

***4.2 Workflow and Data Flow Explanation***

***4.3 Route Management and Page Linking***

***5. Setup Process***

***5.1 Installing & Configuring the Backend***

***5.2 Frontend Environment Initialization***

***5.3 Setting Up Environment Variables (.env)***

***6. How to Use the App***

***6.1 User Registration & Login Process***

***6.2 Creating New Tasks***

***6.3 Modifying or Removing Existing Tasks***

***6.4 Task Search & Filter Features***

***6.5 Visual Task Progress Tracking***

***6.6 Logout Flow & Session Termination***

***7. Interface & Visual Screens***

***7.1 Authentication Pages (Login / Signup)***

***7.2 Dashboard View***

***7.3 Task Management UI***

***8. Challenges Faced & Solutions***

***9. Conclusion***

***9.1 Key Learnings***

***9.2 Skills Improved***

***10. References***

***10.1 Tools & Libraries Used***

***10.2 Online Resources***

***11. Appendix***

***11.1 GitHub Repository Link***

***11.2 Demo Video Link***

***11.3 Contact Information***

introduction

This report outlines the creation of a Task Manager Web Application developed using the MERN (MongoDB, Express.js, React, Node.js) stack during my internship. The application is designed to offer an easy-to-use platform for organizing tasks and tracking their progress in real-time.

The system features full user authentication (registration and login), along with capabilities to add, update, delete, and search tasks. A visual progress indicator is also included to help users monitor task completion more effectively, promoting clarity and efficiency.

React.js was utilized for the frontend to deliver a responsive and interactive interface. The backend, built with Express.js and Node.js, connects to a cloud-hosted MongoDB Atlas database to ensure smooth data handling.

This project serves as a reflection of my full-stack development abilities, demonstrating skills in building secure and user-friendly applications. Through this hands-on experience, I’ve worked with RESTful APIs, modular frontend architecture, route protection, and the integration of modern web technologies.

|  |
| --- |
| methodology |
| ****DESIGN SPECIFICATION**** **SOLUTION OVERVIEW:**  The proposed solution involves developing a web-based application titled **"Task Manager"**, built on a scalable and modular structure. The application architecture follows a component-based design using the React framework and adheres to RESTful principles for backend interaction. This ensures that the system remains maintainable, extendable, and efficient.  🔷 **5.1 Frontend Development (React.js)**  The client-side of the application is implemented using **React.js**, leveraging functional components to keep the codebase organized and clean. Key features and tools used include:   * **React Hooks** such as useState and useEffect for managing state and lifecycle events * **React Router DOM** to manage navigation between views * **Form validation** for required fields and password confirmation logic * **Responsive UI** achieved with custom CSS and a simple design structure  🔸 Core UI Components & Their Roles  | **Component** | **Description** | | --- | --- | | **AuthPage** | Manages the login and registration forms | | **TaskForm** | Used to create or update tasks | | **Home** | Shows the task list, progress tracker, and search bar | | **App.js** | Controls routing and user session logic | |

🔷 **5.2 Backend Development (Node.js & Express.js)**

The server-side of the Task Manager application is powered by **Node.js** with **Express.js** as the framework for creating a robust RESTful API. This backend handles business logic, API routing, and secure interaction with the database.

Key technologies and practices include:

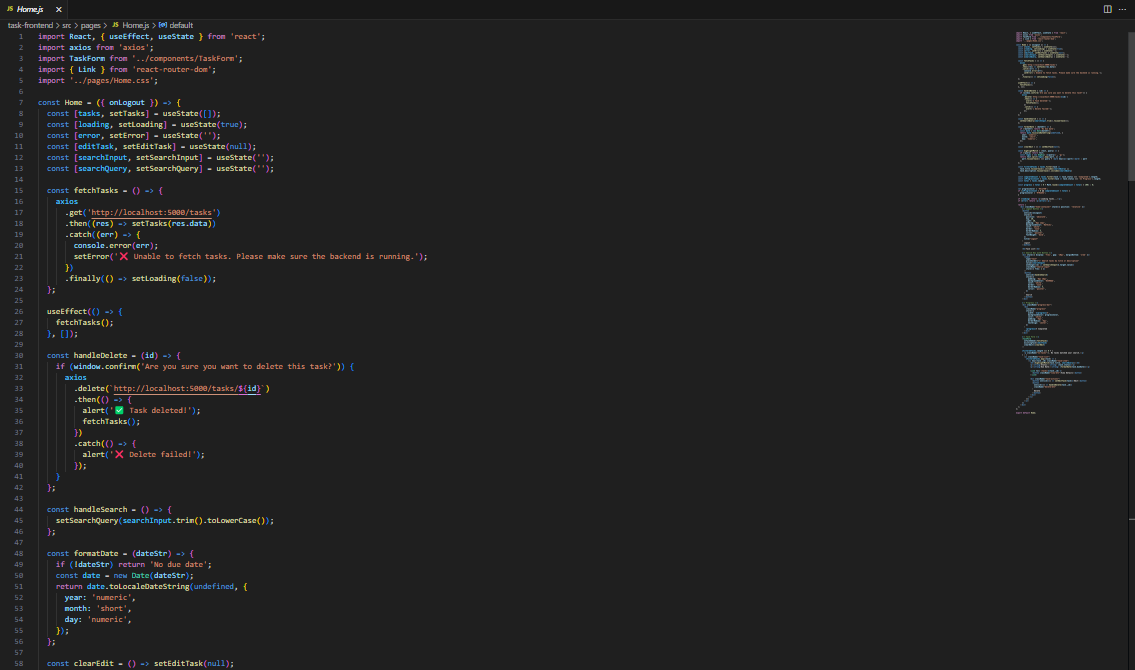
* **Express.js** for building the REST API endpoints
* **MongoDB Atlas** as the cloud-hosted NoSQL database
* **Mongoose** library to define data models and manage database operations
* Clearly defined routes to handle task-related Create, Read, Update, and Delete actions

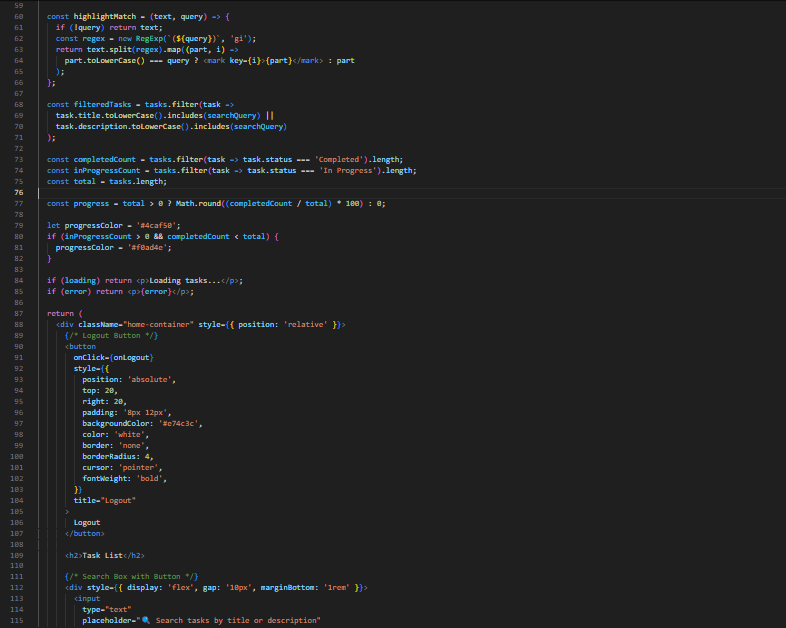
#### 🔸 Sample API Endpoints

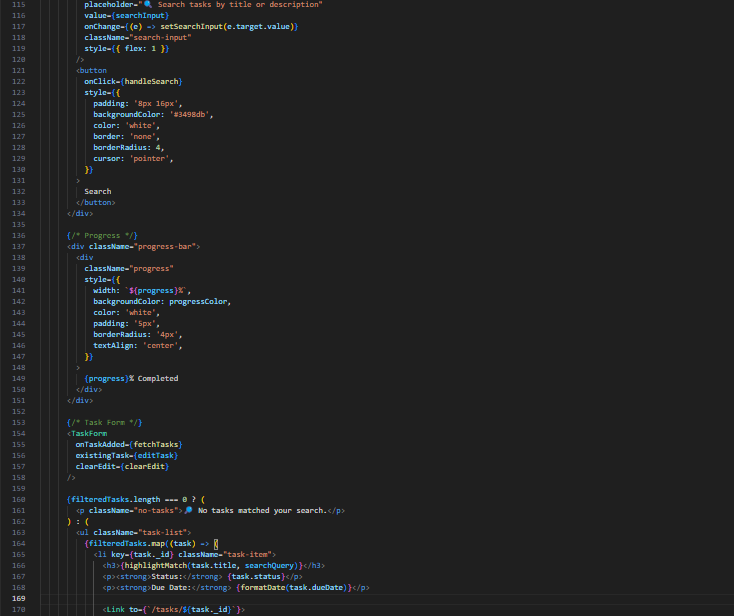
| **HTTP Method** | **Route** | **Purpose** |
| --- | --- | --- |
| GET | /tasks | Retrieve all task records |
| POST | /tasks | Add a new task to the system |
| PUT | /task/:id | Edit an existing task by ID |
| DELETE | /task/:id | Remove a task using its ID |

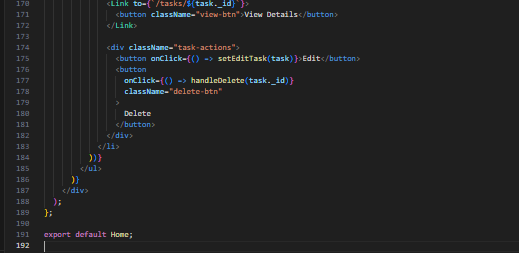
|  |  |
| --- | --- |
| ****Functional Requirements**** These requirements define the primary actions and features the system must support to fulfill user expectations. The **Task Manager Web Application** should deliver the following key functionalities:   1. **User Login & Registration** Secure account creation and access system for users. 2. **Task Handling** Options to add, view, update, and delete tasks with real-time updates. 3. **Search & Filter Tools** Users can quickly locate specific tasks using search keywords or filter criteria. 4. **Progress Visualization** Visual elements such as progress bars to represent task completion clearly. 5. **Client-side Routing** Smooth navigation between different sections of the app using route management.  ****Problem Statement**** In today’s digital landscape, managing daily tasks and tracking productivity remains a common challenge. Many individuals and teams either use outdated methods like handwritten lists and spreadsheets, or find modern task management apps unnecessarily complex for their basic needs.  There's a clear demand for a **simple, efficient, and accessible task management tool** that allows users to:   * Quickly create and organize tasks * Get a visual overview of task progress * Filter and search through tasks effortlessly * Ensure their data remains protected through authentication * Interact with a modern, responsive user interface   This project addresses these issues by delivering a streamlined **full-stack Task Manager** built on the MERN stack (MongoDB, Express.js, React.js, Node.js). It provides core features like login/signup, task operations, progress tracking, and real-time search — all wrapped in a clean and minimal design. This solution is especially ideal for individuals or small teams seeking a fast, user-friendly way to manage their work without unnecessary complexity. ****Documentation**** Managing daily tasks can become overwhelming when users don't have access to straightforward and efficient tools. While numerous task management applications exist, many are either overly complicated or fail to offer the core features that users truly need. There is a clear demand for a minimal, user-centric web solution that enables task organization, progress tracking, and focused workflow—all within a simplified digital interface.  An ideal project should be supported by structured documentation presented in the form of a formal project report. The report typically includes the following sections:   * **Abstract** – A brief summary of the project’s purpose and outcomes. * **Table of Contents** – A navigational guide to all sections of the document. * **Project Scope** – Defines the boundaries and goals of the application. * **Functional Requirements** – Details of core system functionalities. * **Screenshots & User Interface** – Visuals representing the system's layout and key screens. | |
|  |  |
| Screenshots:  *Interface:*  *Frontend*  *Component ( TaskForm.js / TaskForm.css )* |
|  |

***(Home.js / Home.css)***

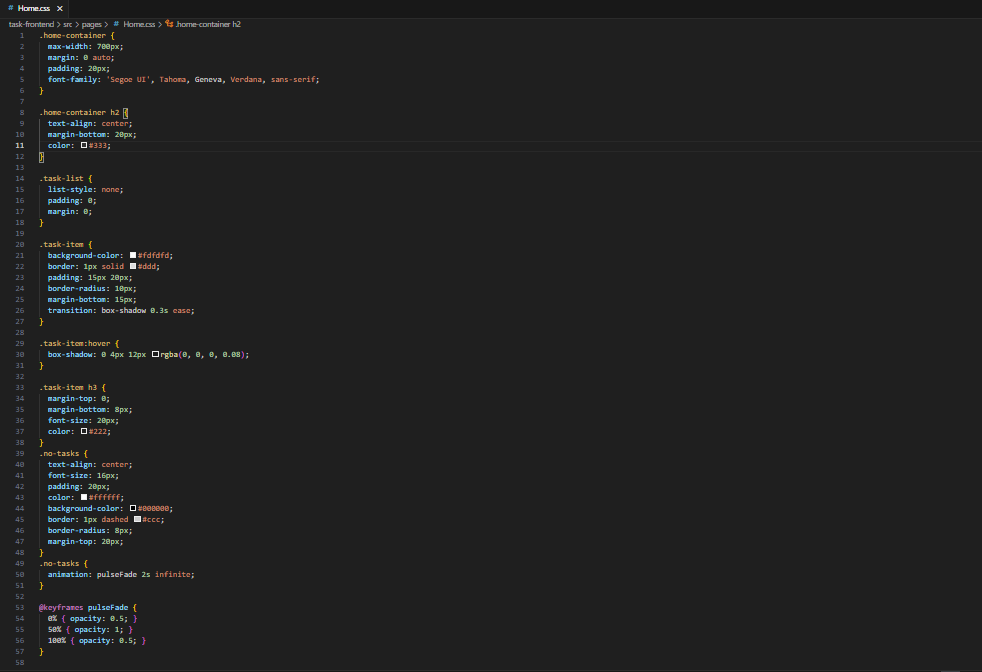
******

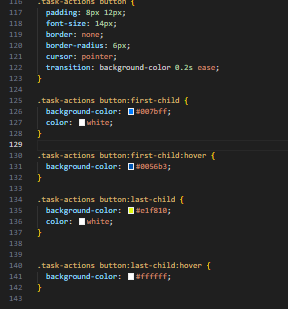


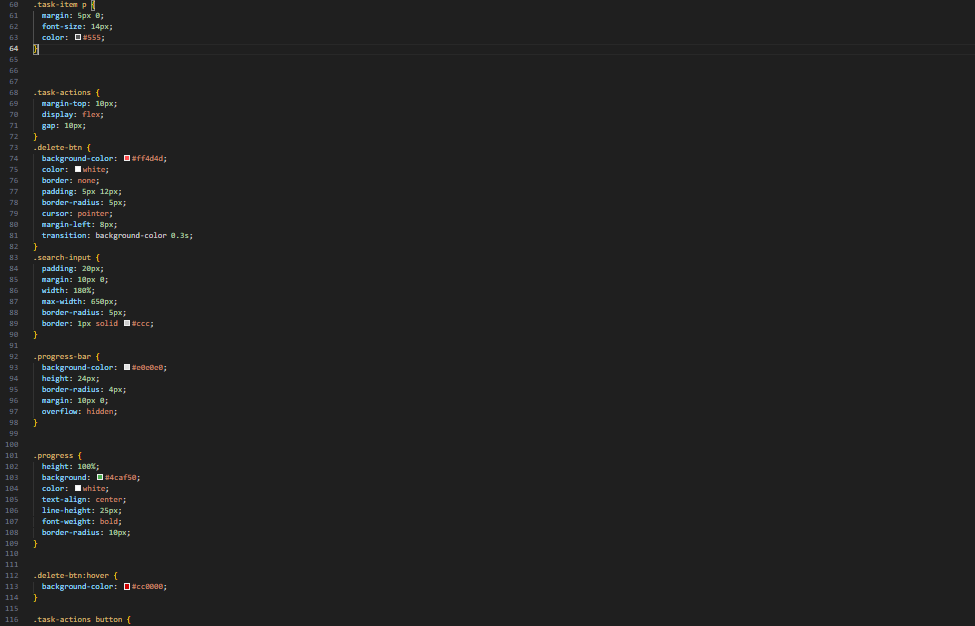




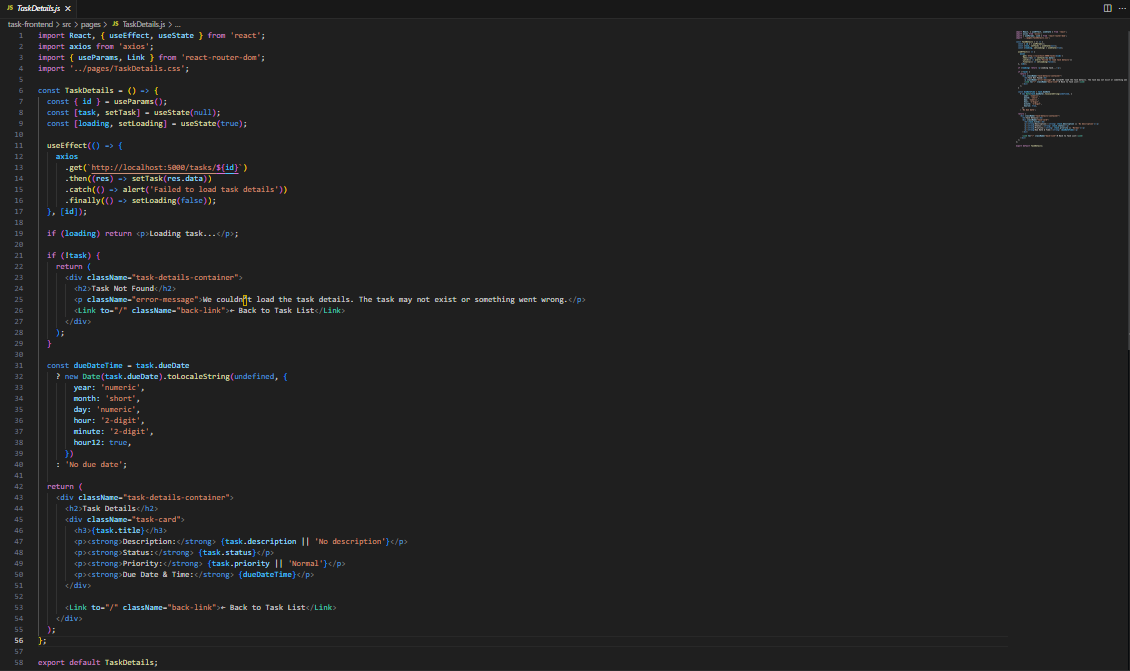
***Home.css***

******

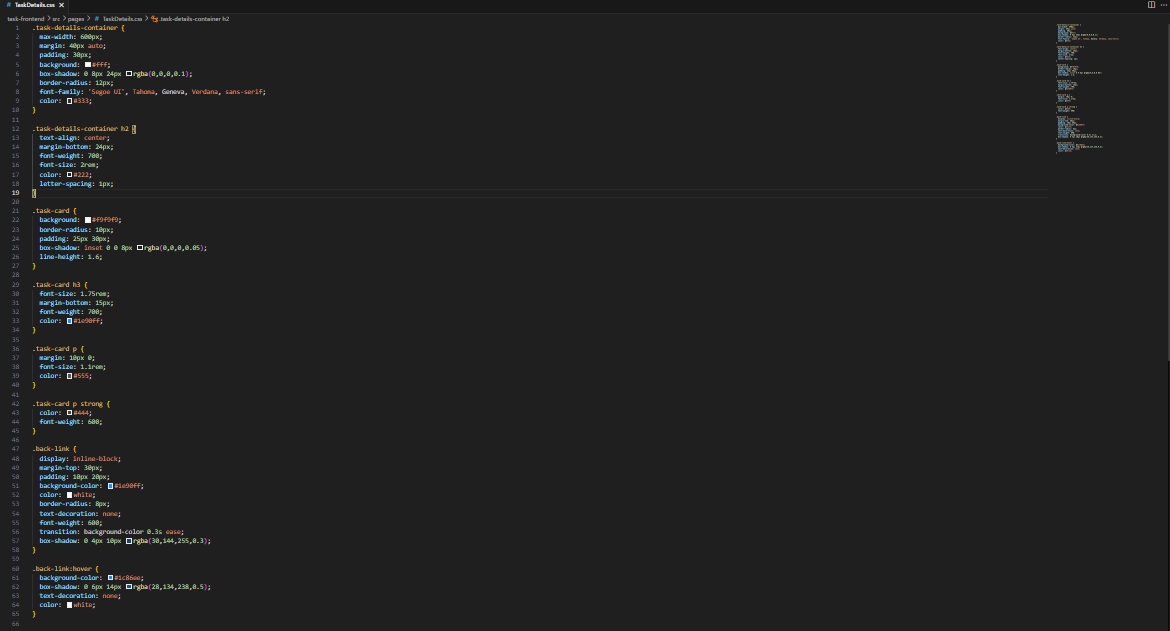




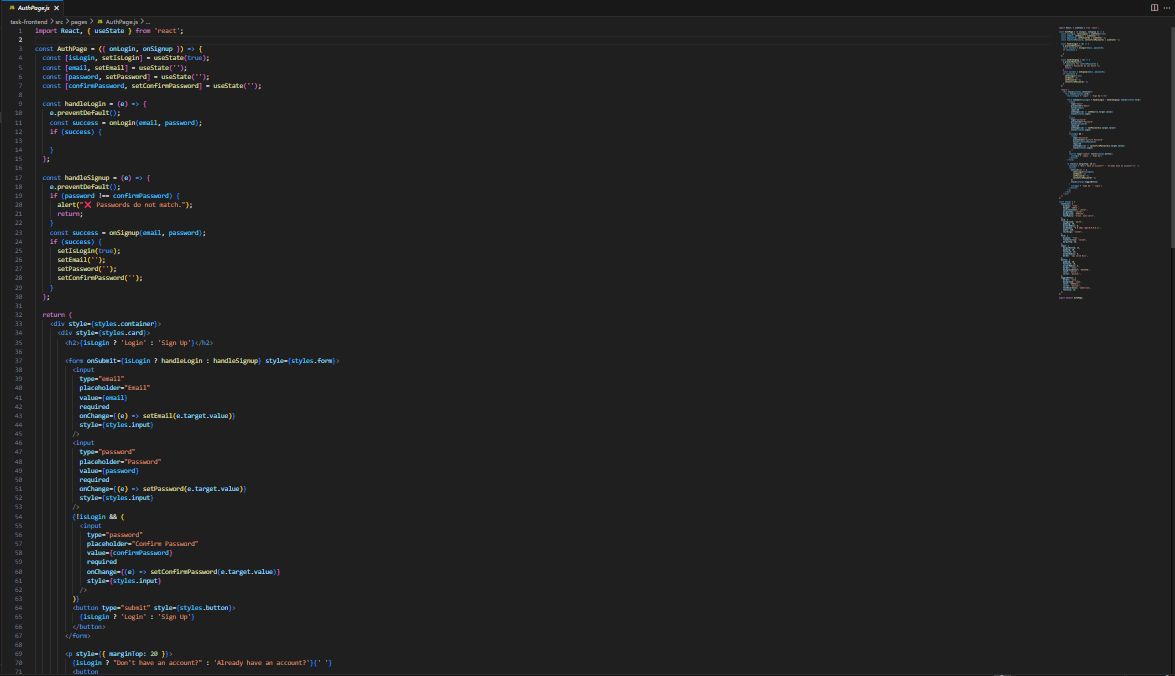
***PAGE ( Taskdetails.js / Task details.css ):***

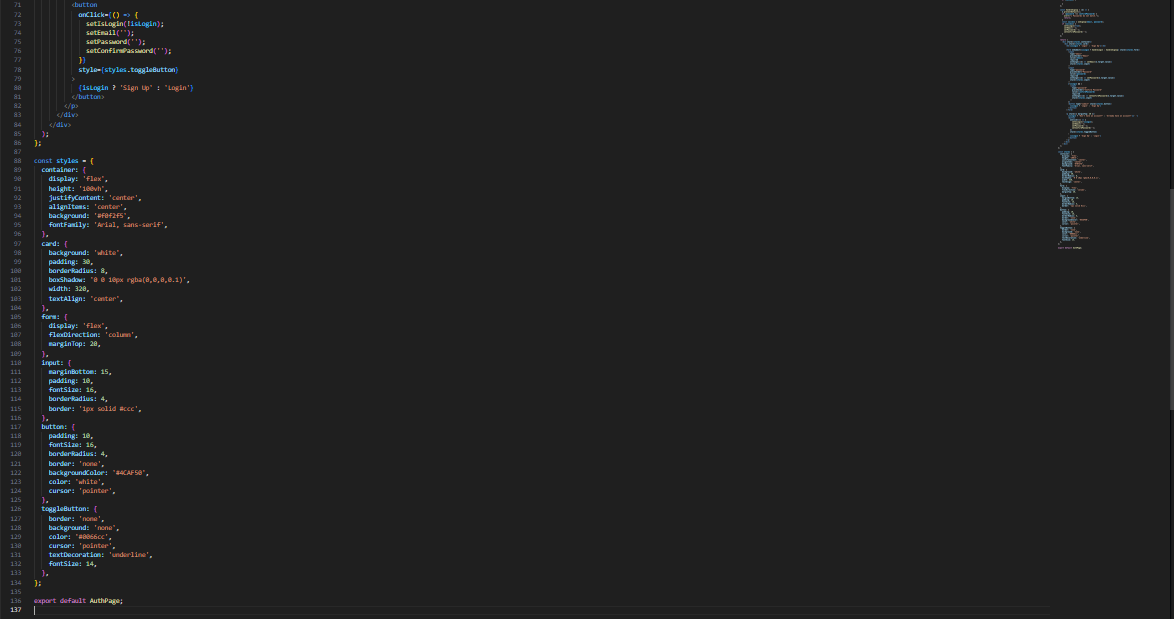


***Taskdetails.css***

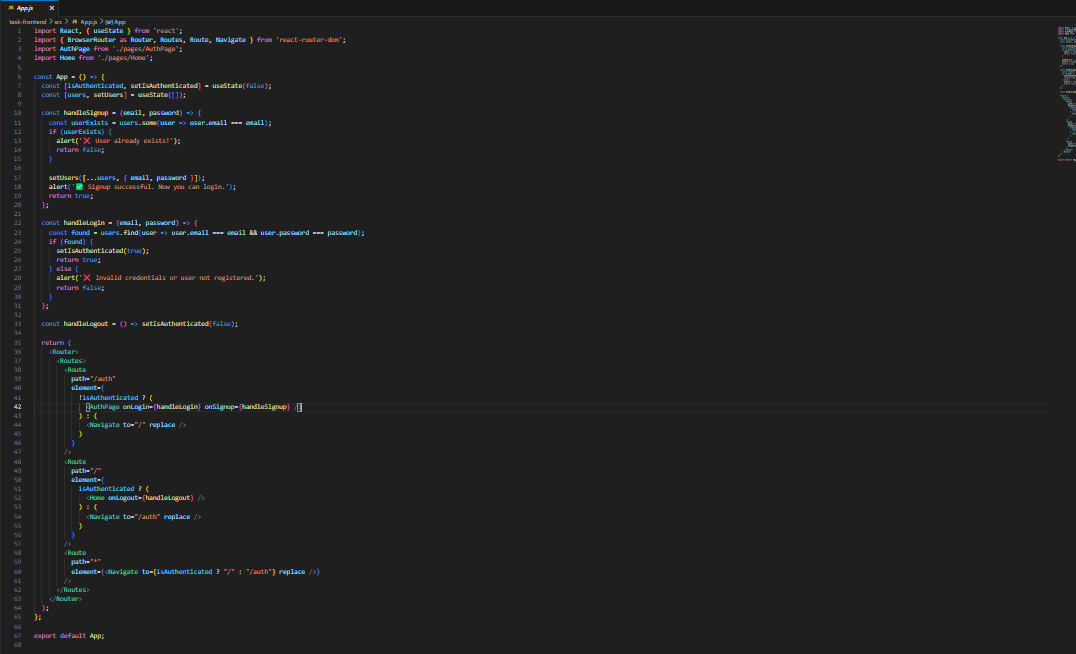
******

***PAGE ( AuthPage.js ):***

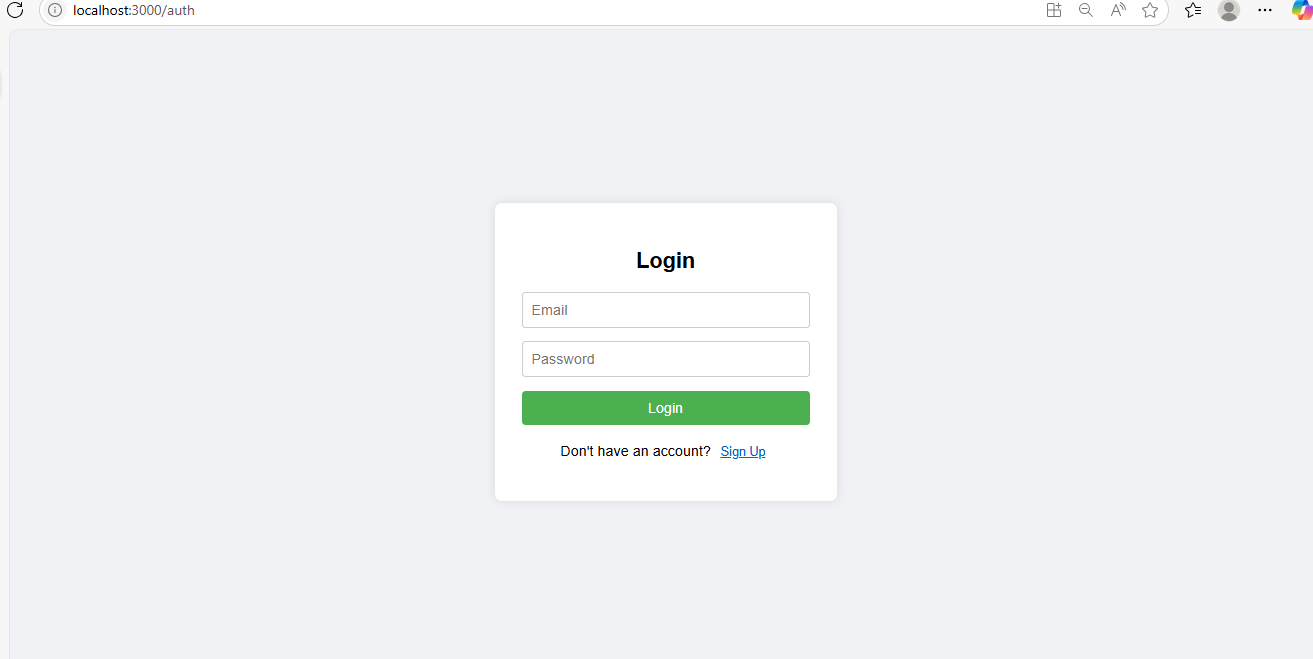


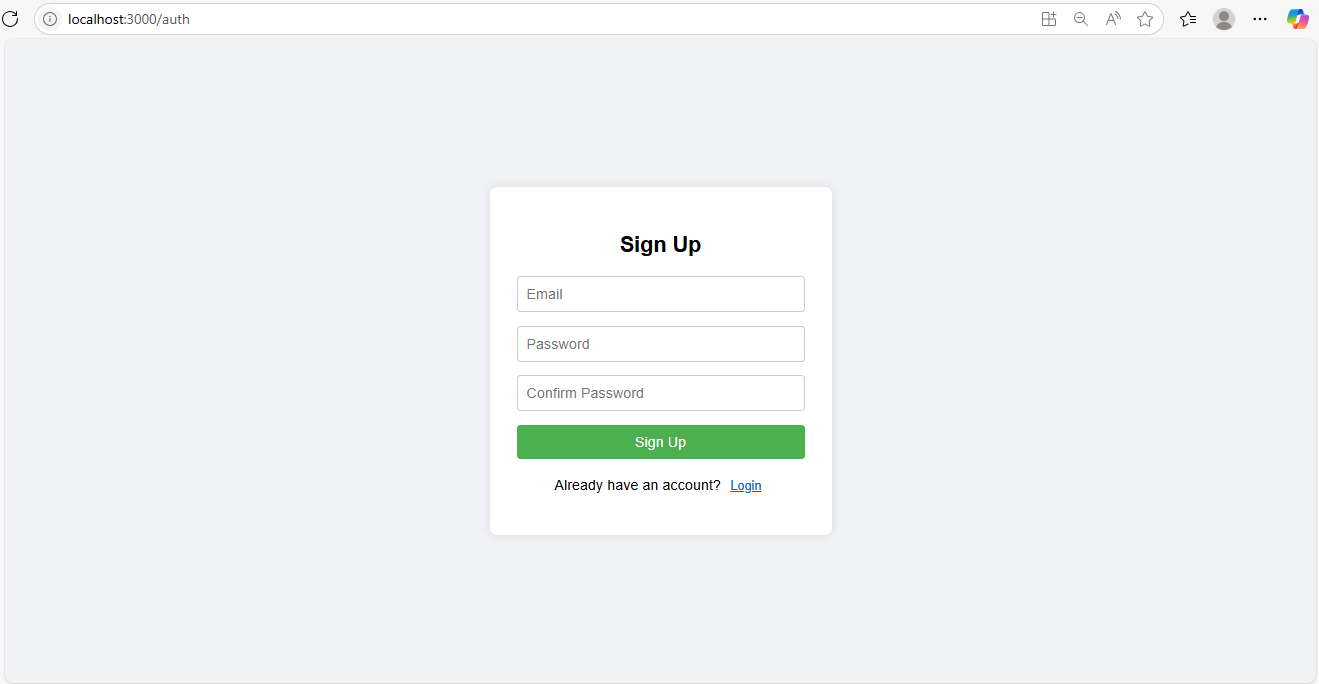


***App.js:***

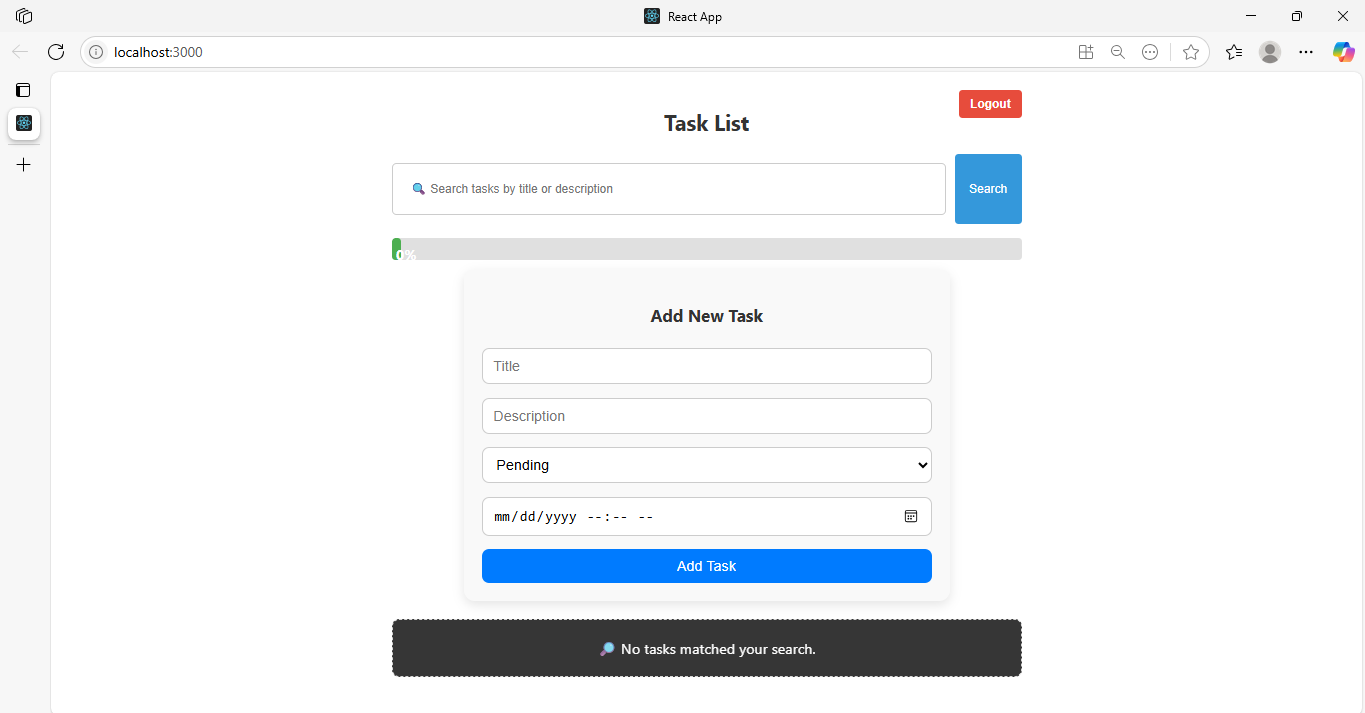


***Auth Page :***

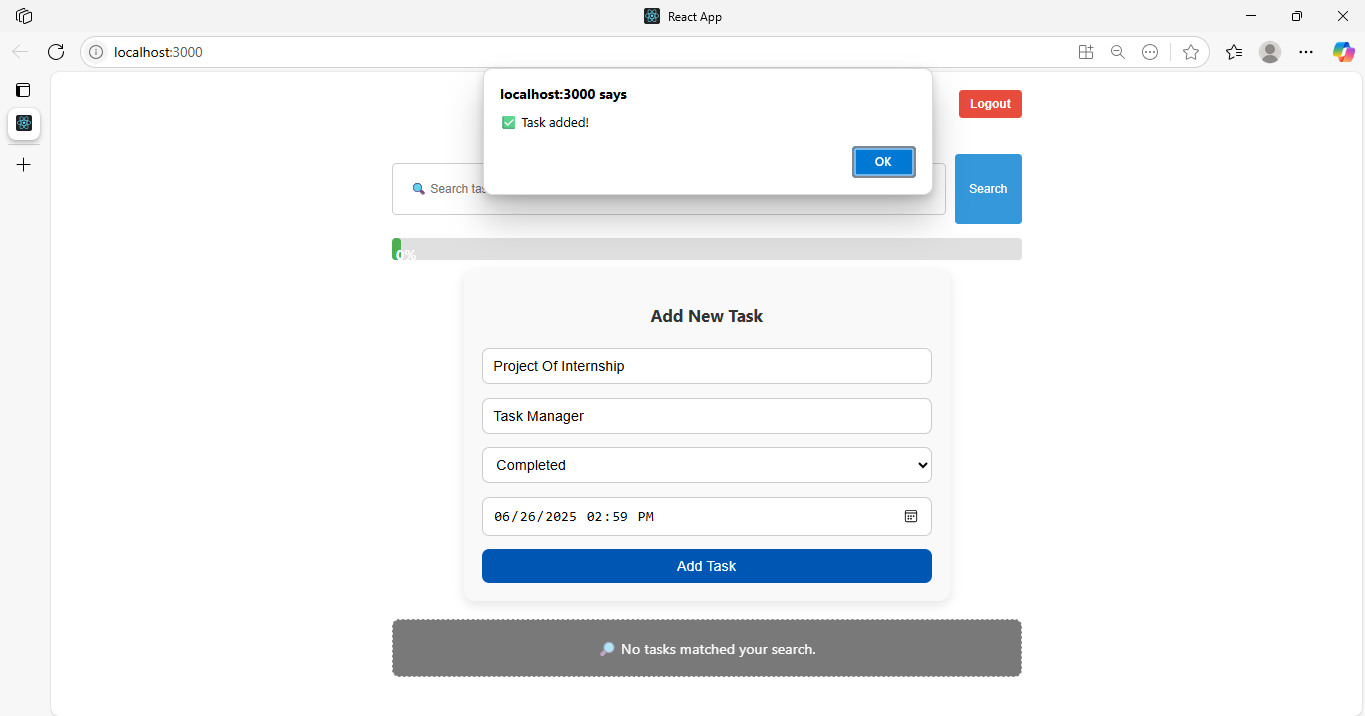


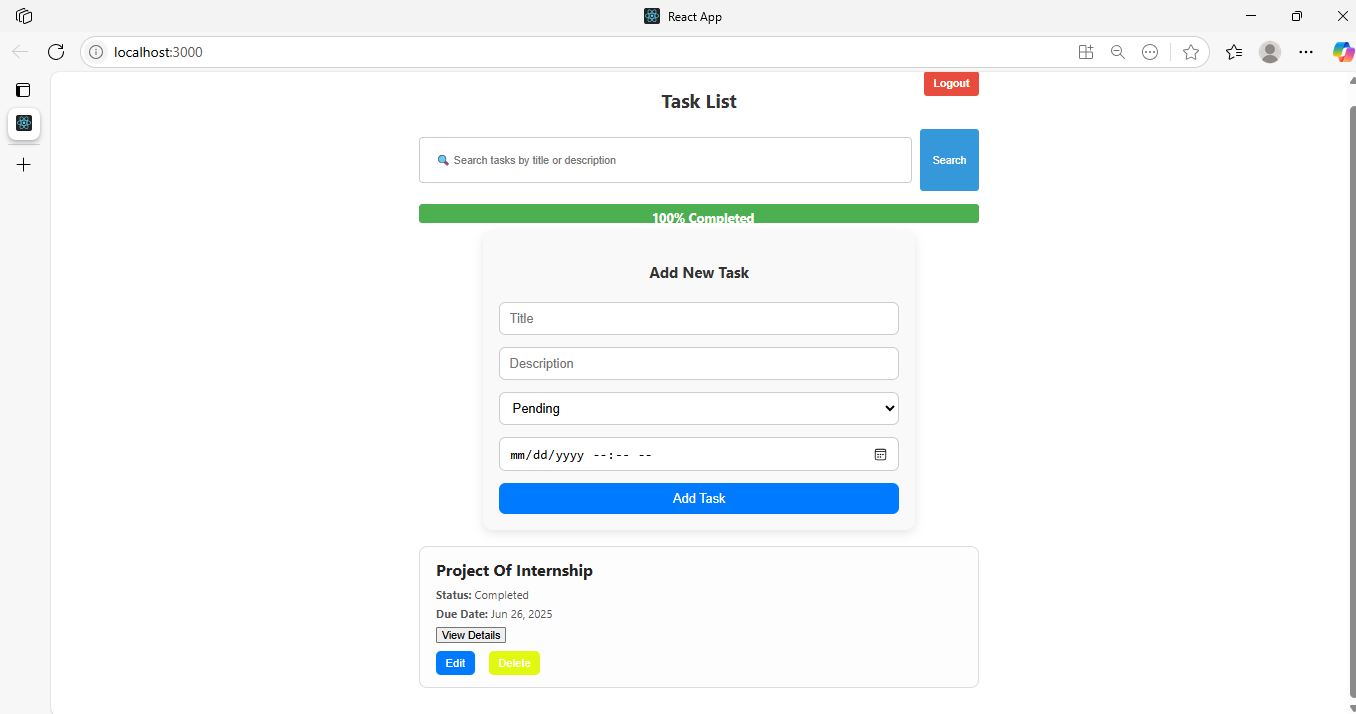


***Home:***

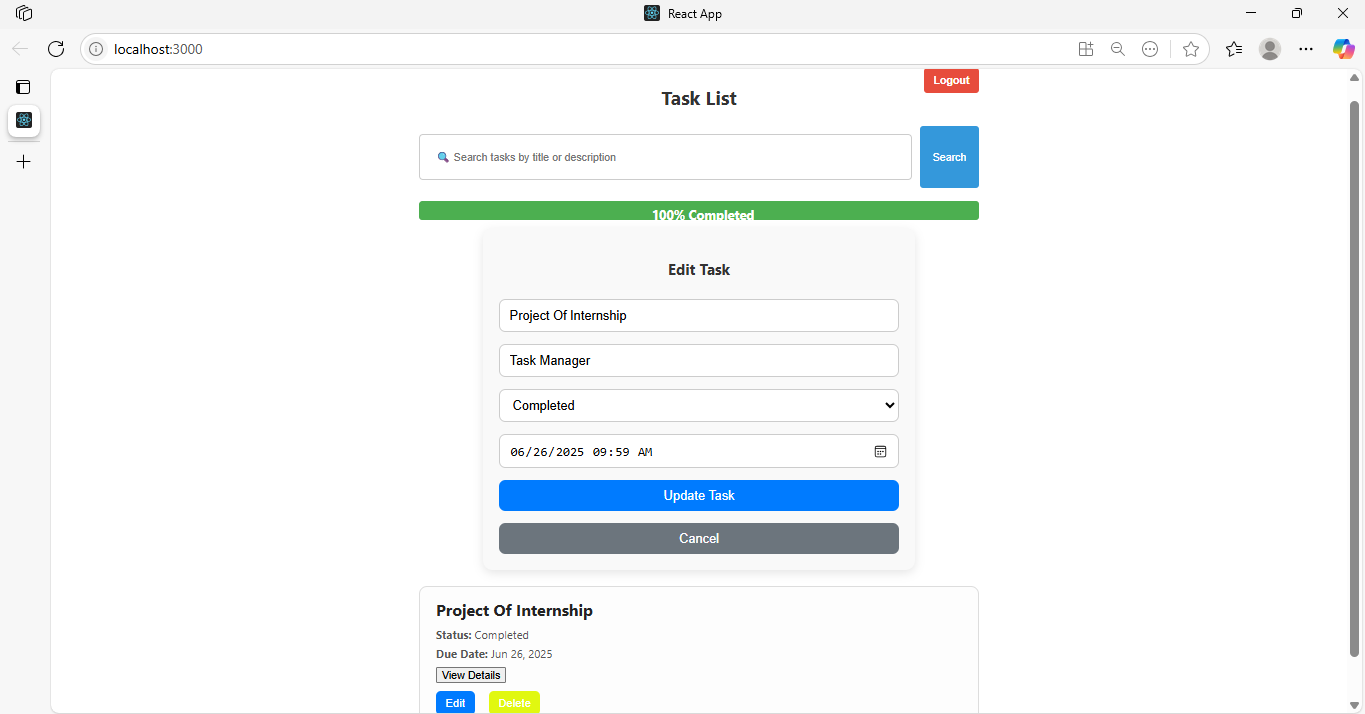


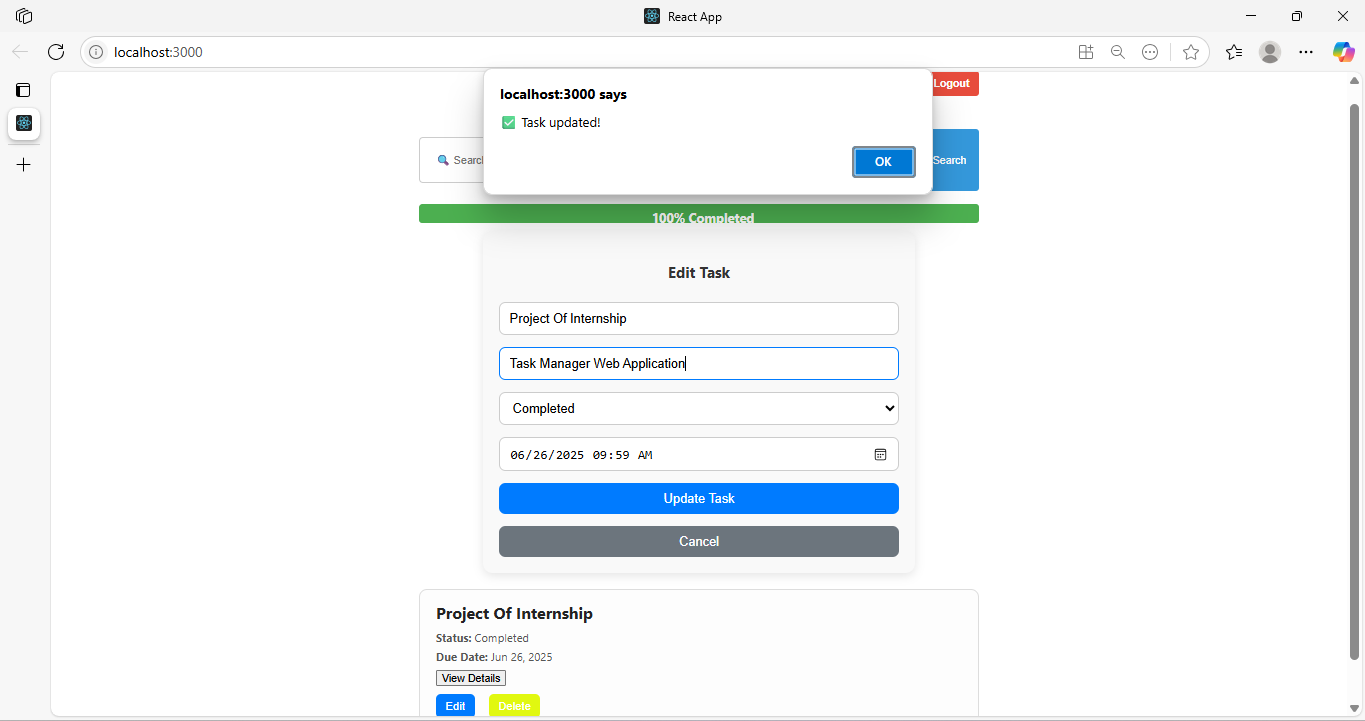
***Task Added with Details:***

******

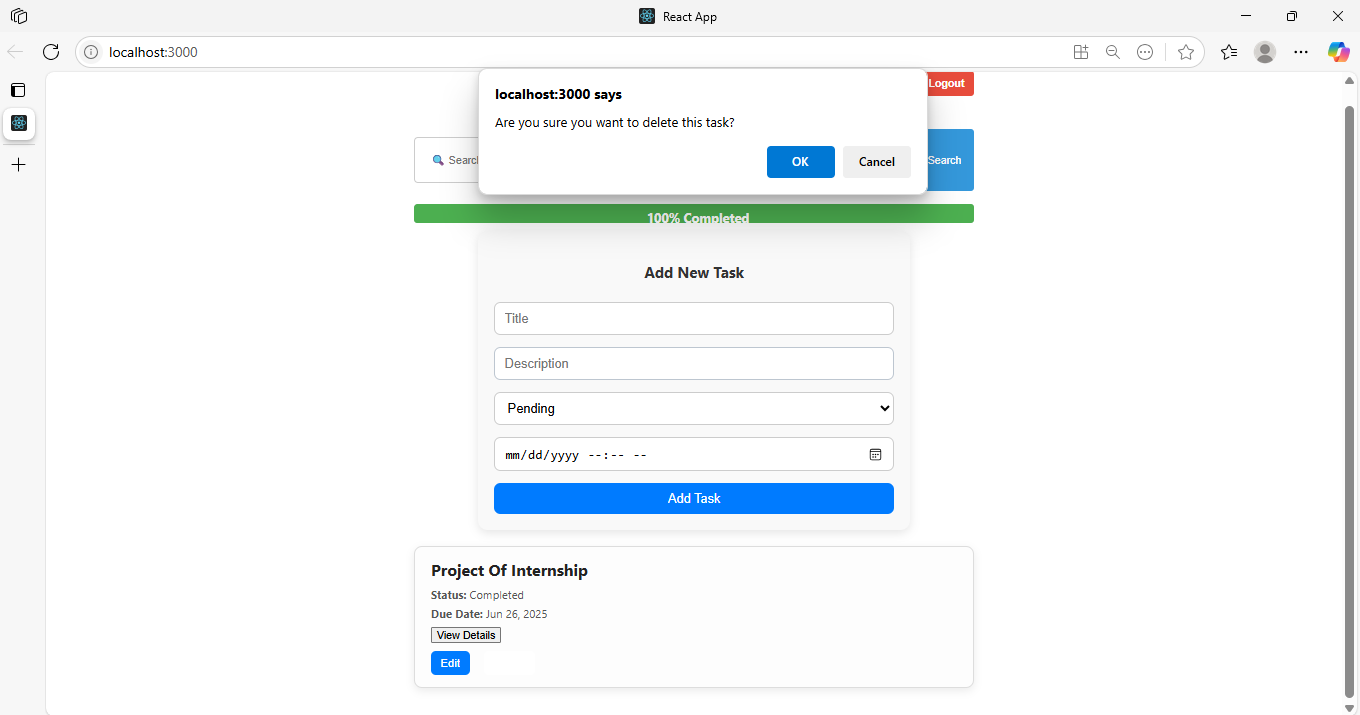


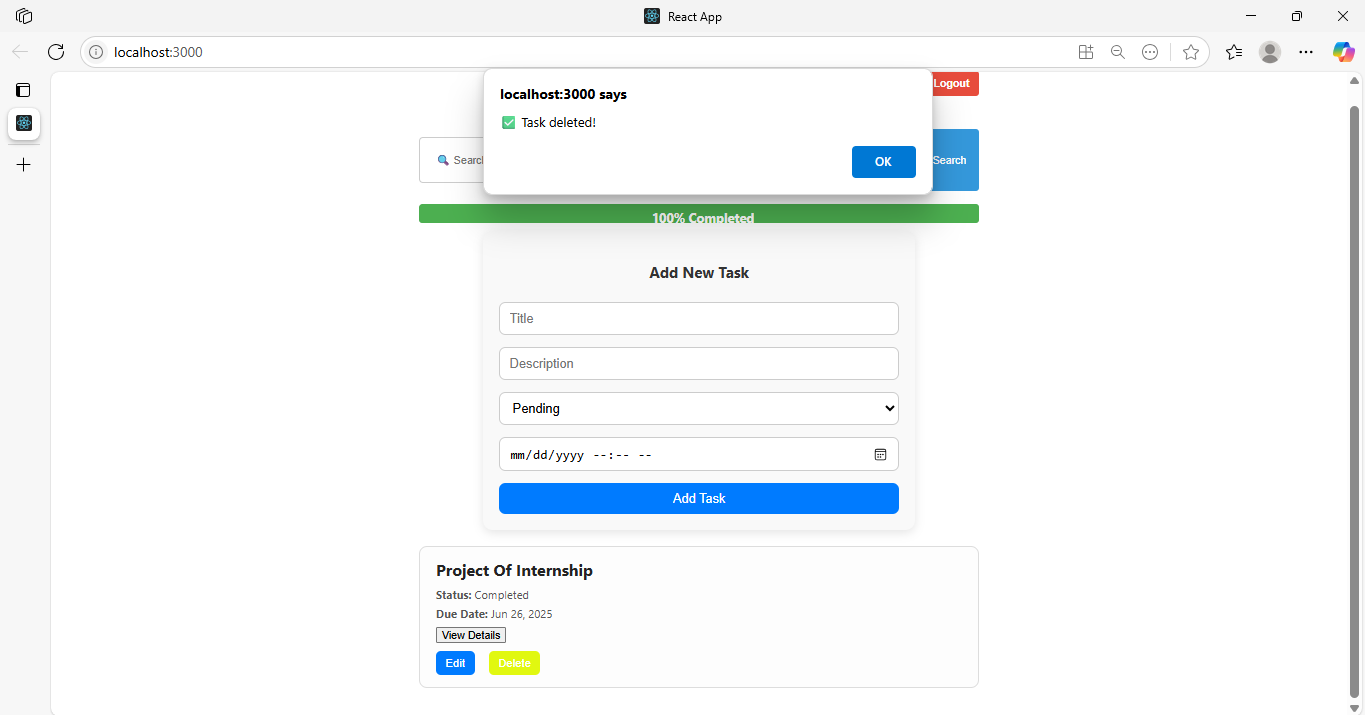
***Then Edit Task:***

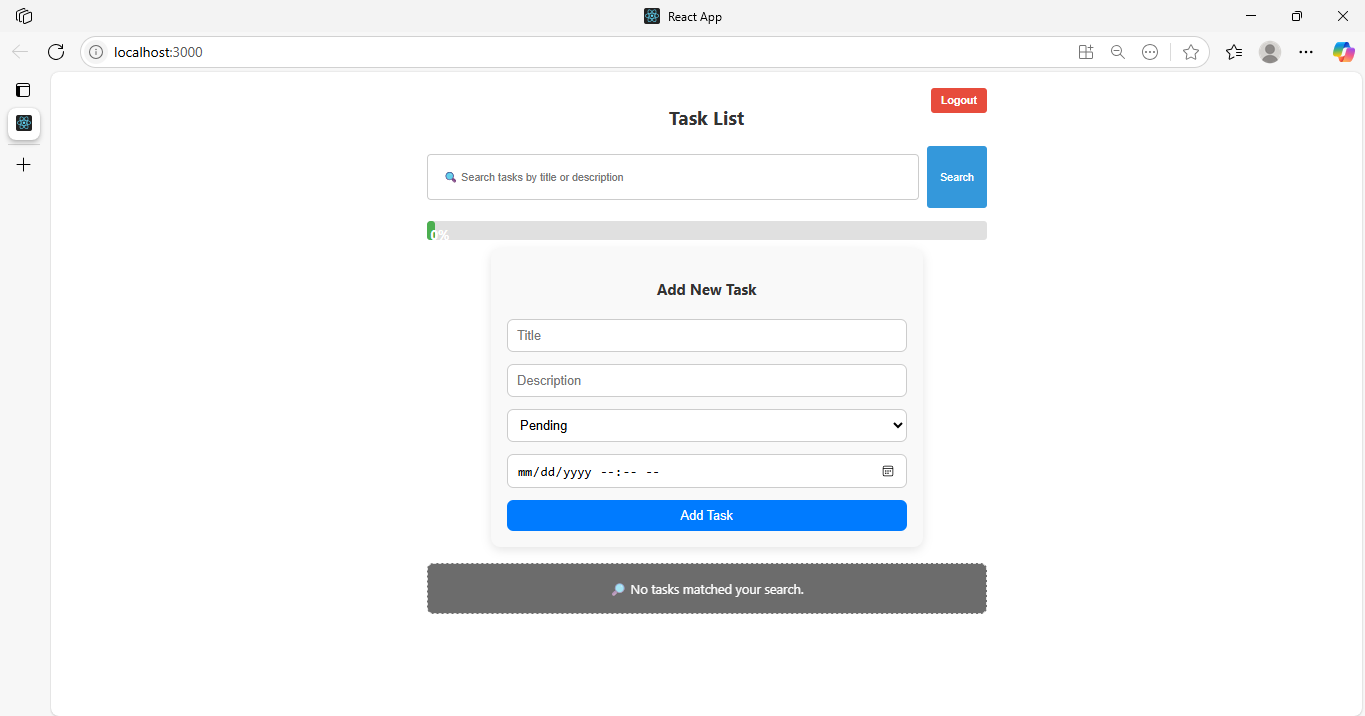




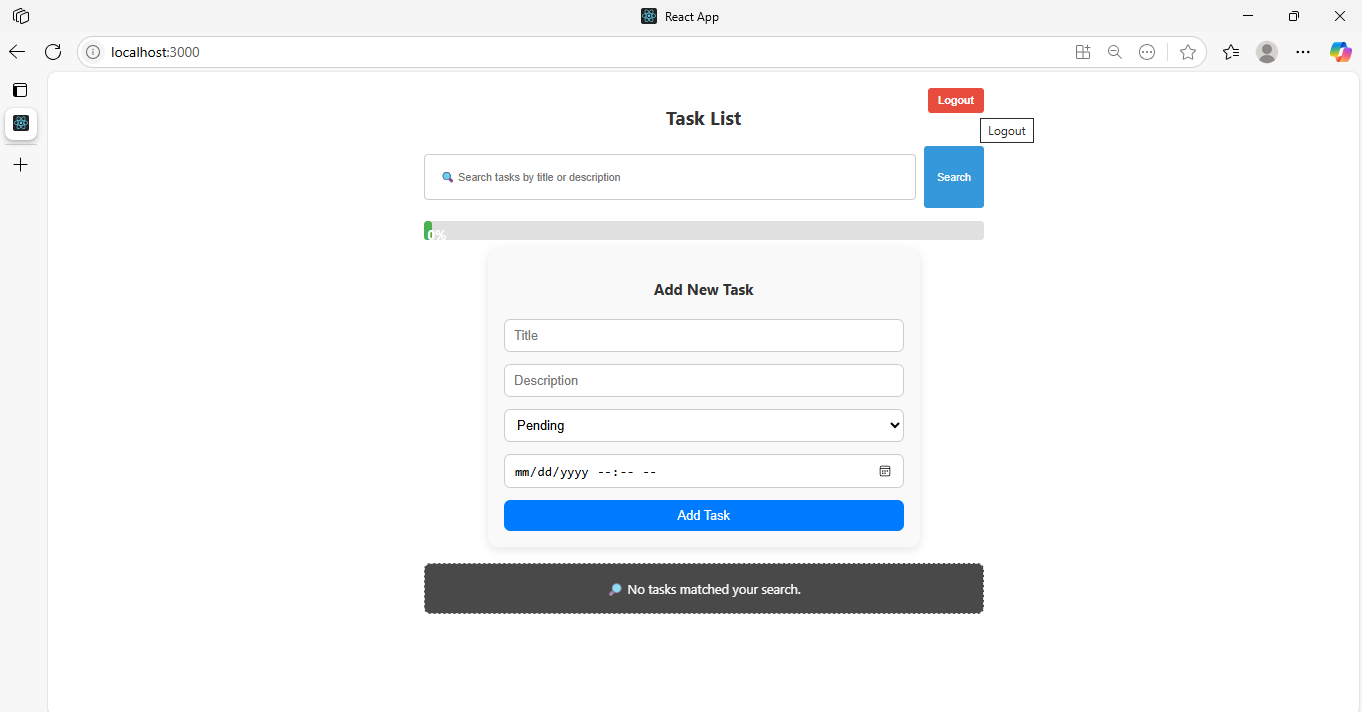
***Delete Task:***





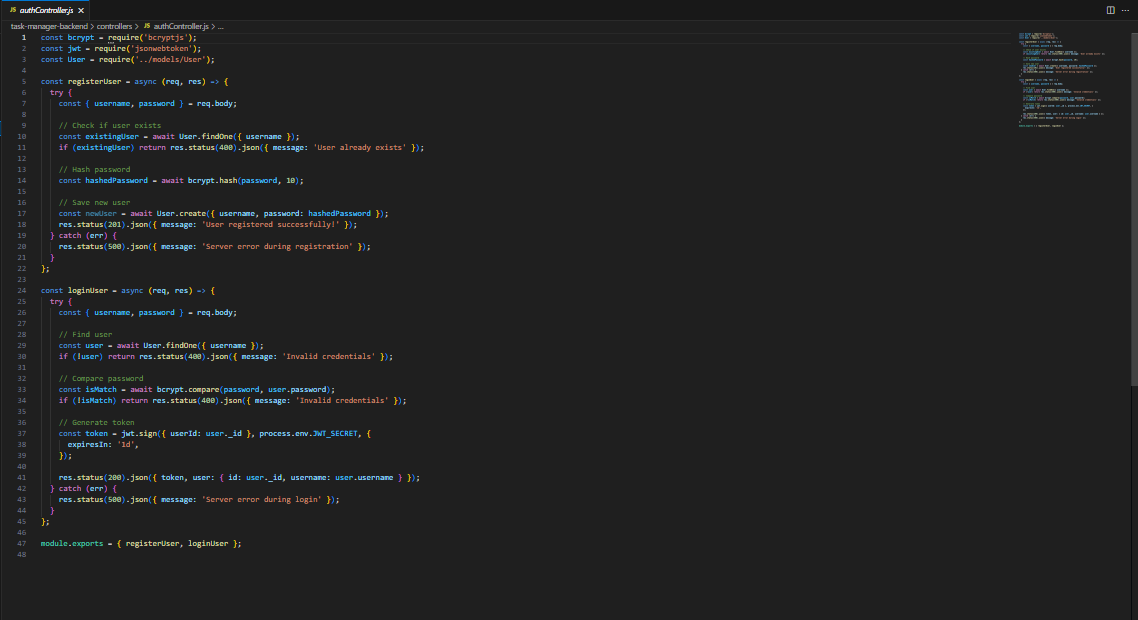


***Then Logout:***

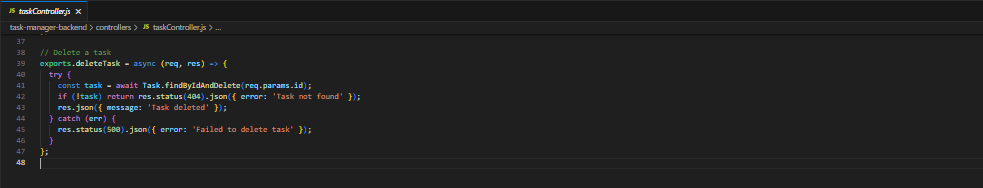


***BACKEND Screenshots***

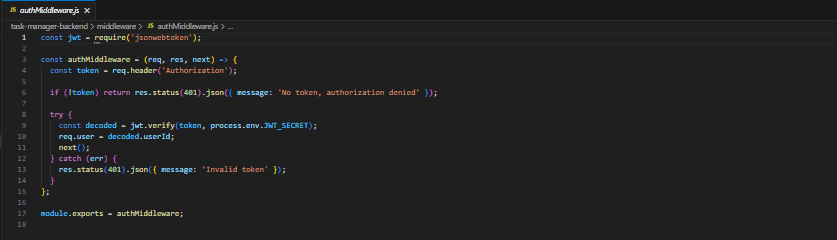
***AuthController.js***

******

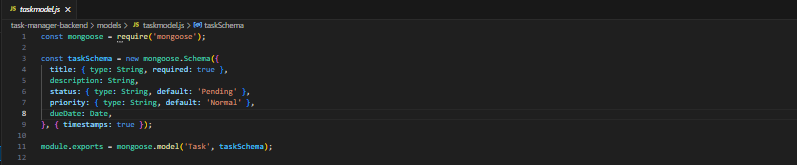
***Taskcontroller.js***

******

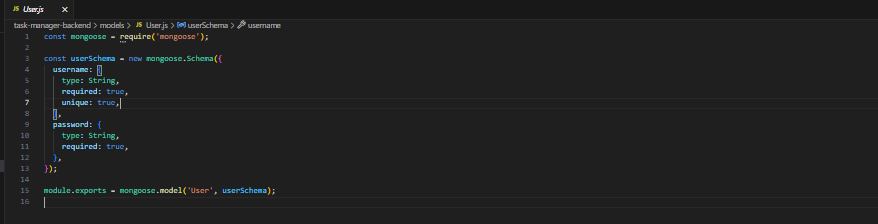
***authMiddleware.js***



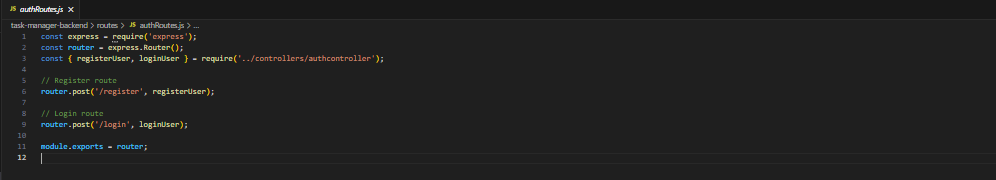
***taskModel.js***



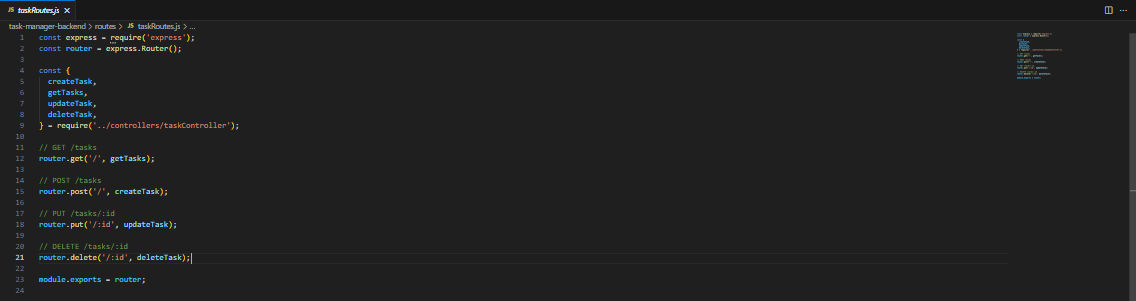
***UserModel.js***



***AuthRoutes.js***



***taskRoutes.js***

******

***Thank you!!!***