EFFICIENT-TASK

[task management system-Documentation]

2020/ICT/12
IT-2234

Efficient-Task Project Documentation

Table of Contents

❖ Introduction

Efficient-Task is a comprehensive to-do list application built using the MERN (MongoDB, Express, React, Node) stack, providing a robust platform for users to manage their tasks efficiently. The application is designed to cater to individuals seeking a seamless and organized approach to task management, allowing them to register, log in, and maintain their to-do lists with ease.

The primary goal of Efficient-Task is to enhance productivity by offering a user-friendly interface and a suite of features that streamline the task management process. Users can create new tasks, update existing ones, mark tasks as completed, and delete tasks when necessary.

This project was developed as part of a university assessment, demonstrating the practical application of modern web development technologies in solving real-world problems. By leveraging the strengths of the MERN stack, Efficient-Task showcases the potential of these technologies to build scalable, responsive, and efficient web applications.

- **Key Features:**

  User Authentication: Secure user registration and login functionality to ensure personalized task management.

  Task Management: Comprehensive features for adding, updating, completing, and deleting tasks.

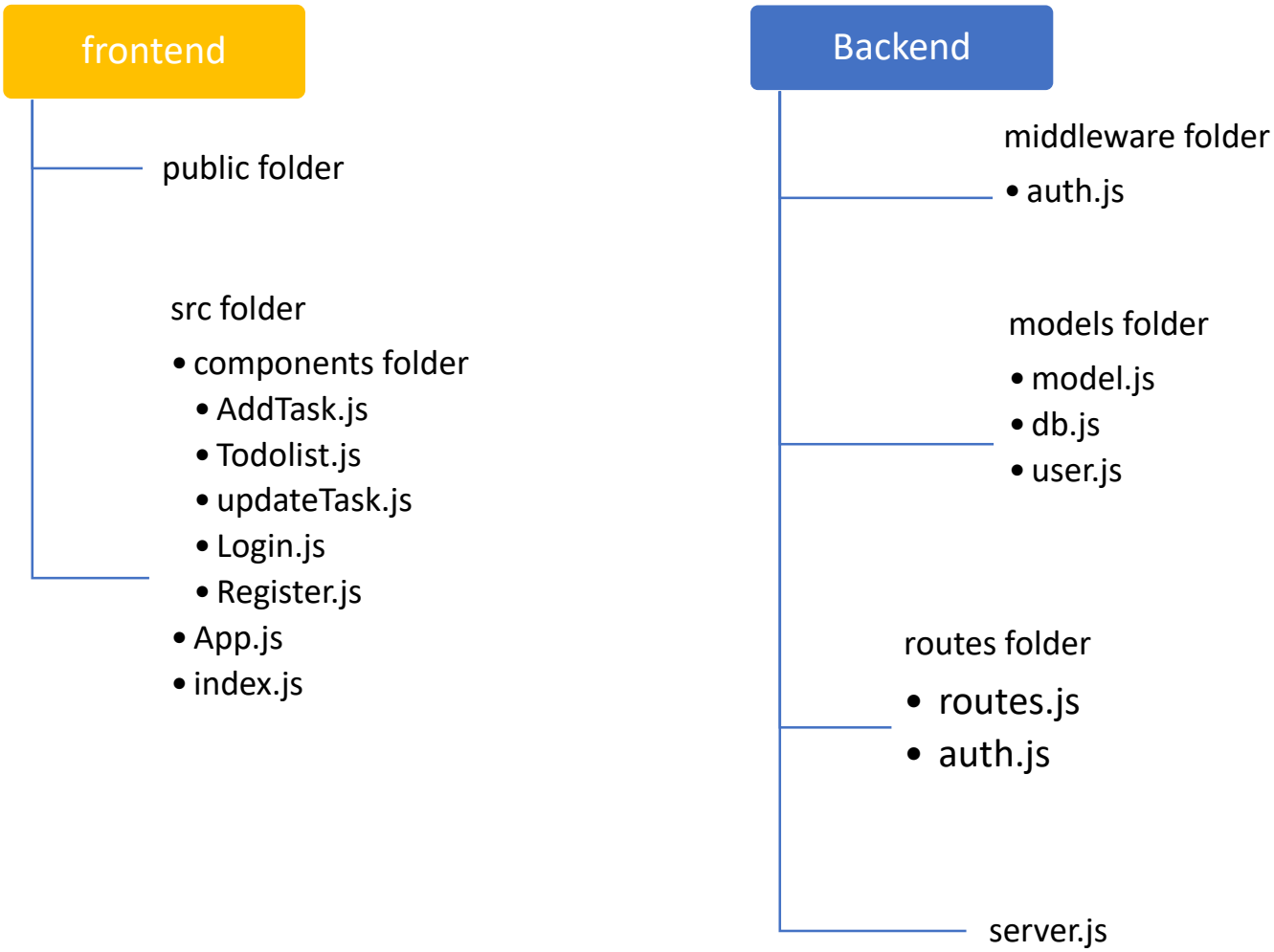
Efficient-Task not only serves as a valuable tool for managing daily tasks but also underscores the capabilities of the MERN stack in developing modern web applications. Through this project, I have gained hands-on experience and deepened my understanding of full-stack development, contributing significantly to my growth and skills in the field of web development.

❖ Project Architecture

The project is divided into two main parts: the frontend and the backend.

1) Frontend
   Framework: React
2) Backend
   Framework: Node.js with Express
3) Database: MongoDB

➢ Folder Structure:



❖ API Documentation

- User Routes

  - Register a user

URL: /api/auth/register
Method: POST
Description: Registers a new user.
Request Body:
json
{
"username": "string",
"password": "string"
}
Response: Redirected to `/login` on successful registration or in console (Error: “Error registering”)

  - Login a User

URL: /api/auth/login
Method: POST
Description: Authenticates a user.
Request Body:
json
{
"username": "string",
"password": "string"
}
Response: Redirected to `/` (todoList page on successful registration or in console (Error: “Error logging in”)

- Task routes
  - Add a Task

URL: /api/tasks/
Method: POST
Description: Adds a new task.
Request Body:
json
{
"todo": "string"
}
Response:
json
{
"task": { "id": "string", "todo": "string", "completed": false }
}

- Get All Tasks

URL: /api/tasks
Method: GET
Description: Retrieves all tasks for the logged-in user.
Response:
json
```
{
  "tasks": [
    { "id": "string", "todo": "string", "completed": false },
    ...
  ]
}
```

- Update a Task

URL: /api/tasks/:id
Method: PUT
Description: Updates an existing task.
Request Body:
json
```
{
  "todo": "string",
  "completed": "boolean"
}
```
Response:
json
```
{
  "task": { "id": "string", "todo": "string", "completed": "boolean" }
}
```

- Delete a Task

URL: /api/tasks/:id
Method: DELETE
Description: Deletes a task.
Response:
json
```
{
  "task": { "id": "string" }
}
```

➢ System Functionalities

- Individual Account Management: Users can log in to their individual accounts and manage their tasks.
- Add Task: Users can add new tasks by typing the task description in the input field and clicking the "Add Task" button. This triggers an onclick event to add the task.
- Update Task: Users can update existing tasks by clicking the edit mark next to the task. This triggers an onclick event to edit the task

- Mark Task as Completed: Users can mark tasks as completed by clicking the added task, and This is visually represented with a strikethrough line and a checkmark, triggered by an onclick event.
- Delete Task: Users can delete tasks by clicking the delete mark next to the edit mark. This triggers an onclick event to delete the task.

❖ Usage Instructions

Prerequisites

1. Node.js and npm installed
2. MongoDB installed and running

Installation

1. Clone the repository:

   git clone https://github.com/SGopinath89/IT22342024EfficientTask

2. Navigate to the project directory and install dependencies for both frontend and backend:

Open 2 new terminals

In one of the terminal, do the following steps

I. cd Efficient-Task
II. cd frontend
III. npm install

in another terminal,do the following steps

I. cd Efficient-Task
II. cd backend
III. npm install

Running the Application

Start the backend server:

I. cd backend
II. node server.js

Start the frontend development server:

I. cd frontend
II. npm start
III. Open your browser and navigate to http://localhost:3000 to view the application.

❖ Demo Presentation

I've screen recorded my application, and uploaded here, you can click this link to see

https://drive.google.com/file/d/1Rvxblq30_93bThgIBFDoK2RqOvn2GSHE/view?usp=sharing

❖ Development Process

Initial Setup

- Initialized the project structure with separate frontend and backend folders.
- Set up MongoDB connection using Mongoose.

Building the Backend

- Created user authentication using JWT for secure login and registration.
- Implemented task management endpoints (add, update, delete, and fetch tasks).

Building the Frontend

- Designed and implemented the user interface using React
- Added styles to the pages with CSS
- Connected frontend components to backend APIs.

❖ Challenges Faced

- Learning Curve with MERN Stack: As a beginner in MERN stack development, there was a significant learning curve in understanding how MongoDB, Express, React, and Node.js work together.

- Authentication Implementation: Implementing user authentication (login and registration) posed challenges, especially ensuring secure token management and integration with frontend components.

- Handling State Management: Managing state across React components, especially when updating and deleting tasks, required understanding state lifecycles and asynchronous operations.

- Debugging API Requests: Troubleshooting API requests and responses between frontend and backend was challenging, particularly handling errors like 404 (Not Found) and ensuring proper data flow.

- Deployment and Environment Setup: Setting up the development environment, configuring database connections, and deploying the application posed initial setup challenges.

❖ Conclusion

In conclusion, developing the "Efficient-Task" project using the MERN stack has been a valuable learning experience for me as a beginner. This project aimed to create a simple yet functional task management system, integrating MongoDB for database management, Express for server-side logic, React for the front-end interface, and Node.js for server-side scripting. Throughout the development process, I encountered various challenges, such as implementing user authentication, managing state across components, and ensuring seamless integration between front-end and back-end functionalities.

As a novice in MERN stack development, this project has provided me with foundational knowledge and hands-on experience in building full-stack applications. While the current system serves basic functionalities like task creation, update, deletion, and user authentication, it lays a strong groundwork for future enhancements. Potential future improvements include adding more features like task prioritization, reminders, user profiles, and collaboration capabilities to make the system more robust and user-friendly.

I sincerely hope that this project meets the expectations, despite any shortcomings or limitations. I appreciate the opportunity to learn and grow through this project, and I am eager to apply my newfound skills to tackle more complex challenges in future endeavors.