TASK MANAGEMENT

This is a Task Management, a Full Stack Project built using MERN Stack.

The technologies used are :
1. ReactJS for the Frontend.
2. NodeJS and ExpressJS as Backend Framework (Node v16.17.0)
3. MongoDB as Database.

Getting Started

Requirements
- NodeJS installed on your machine
  - Node Version used is 16.17.0
- npm package manager (comes with node)
- A text editor or IDE of your choice (VS Code recommended)

Installation
- First of all navigate to "client" folder for Frontend code.
- Navigate to "server" folder for Backend code.

To get started you need to install all dependencies by running: npm i (in both the respective terminals)

TO RUN PROJECT
- start both Frontend and Backend using "npm start" command.

ASSUMPTIONS
- No majoe assumptions were taken in th Development of the project.
- Completion Status of the newly created task is taken as not completed
- Task could be completed using the complete button provided on TaskList form.
- It is assumed that, once the task is completed, status to be changes as Completed and then Complete and Edit buttons are disabled assuming that we cannot edit the task after its completion

VALIDATIONS
- All the Validations defined in the assignment document were kept in mind during the Developmentof the project.
- Any of the Title, Description and Due Date field cannot be empty. (All of them are required)
- Due Date cannot accept dates of the past, a future date needs to be provided by the user orelse it will generate and error.

ERROR HANDLING
- Proper Error handling is being carried out in both Framework and Backend with proper error messages whereever required.

Challenges & Solutions:

- I am more confident on the Backend Development so I was able to design the backend of the project in an easier way.

- I am not that much comfortable with Frontend Development so I took more time for designing frontend a nd seeked advice and suggestion from my friends,

they suggested me the resources and the roadmap to design and devlop the frontend of the project.

- I took help from web for R&D of the concepts.

- The project is being designed keeping in mind the Requirements provided in the assignment document.

- I did not came across any major security issue in Development of this project.

But in a more complex application we can make use of following security measures:

- Validate and sanitize all user inputs on both the client and server sides to prevent injection attacks.
- Authentication and authorization of API's using JWT Tokens.
- Implementing CORS policy for cross origin resource sharing.
- Validate and sanitize input for API endpoints.
- Using Bcrypt for hashing the passwords.
- Implement session management and token-based authentication.
- Securing routes in both Frontend and Backend.
- Use strong passwords and encryption for database access.
- Apply the principle of least privilege for database users.
- Implement proper error handling to avoid exposing sensitive information in error messages.
- Log errors securely without revealing too much information.
- Regularly backup data.
- Implement secure session management techniques.

STEPS

1. The first step was setting up the project structure which includes creating two separate projects one for frontend and another for backend.
   Frontend is in the "client" folder.
   Backend is in the "server" folder.

2. Second was Installation of the appropriate packages/modules/dependencies required in Development f or both the Frontend and Backend.

3. In backend to re-run the server automatically I used the nodemon package provided by npm.

4. Some of the dependencies used in Backend Development are:
   "body-parser": "^1.20.2",
   "cors": "^2.8.5",
   "express": "^4.18.2",
   "mongoose": "^8.0.1",
   "nodemon": "^3.0.1"

5. Some of the dependencies used in Frontend Development are:
   "react": "^18.2.0",
   "react-axios": "^2.0.6",
   "react-dom": "^18.2.0",
   "react-router-dom": "^6.18.0",
   "react-scripts": "5.0.1",
   "web-vitals": "^2.1.4"

API Endpoints

Model
The Model for the Task is following:
    title, description,dueDate,completed
where title, description and dueDate keys are considered to be REQUIRED.

Base URL : http://localhost:5000/api/tasks

Below are the Endpoints of the CRUD API’s :

1. To Create A Task : POST Method
Endpoint - http://localhost:5000/api/tasks

2. To Get list of All Task : GET Method
Endpoint - http://localhost:5000/api/tasks

3. To Get A Task by Id : GET Method
Endpoint - http://localhost:5000/api/tasks/:id (id as a param)

4. To Update A Task : PUT Method
Endpoint - http://localhost:5000/api/tasks/:id (id as a param)

5. To Delete A Task : DELETE Method
Endpoint - http://localhost:5000/api/tasks/:id (id as a param)