MR. Sovon

Winnipeg, MB | +1 (204) 881-1439 | sovon.pac@gmail.com

TODO APP

Important links & credentials:

Admin Login:

Email: admin@admin.com
password: 111111

Paypal:
Sandbox Url: https://sandbox.paypal.com

Business email to see payments
Email: sb-akoa529459416@business.example.com
password: ?^Dgf/53

Personal email to make payments
Email: sb-wolai30285130@personal.example.com
Password: ?^Dgf/B3

For Card purchase use the followings or any demo cards that you want
Card number: 4032036985597853
Expiry date: 10/2026
CVC code: 018

Demo site link: http://ec2-18-118-163-90.us-east-2.compute.amazonaws.com:5000/

This project introduces a comprehensive TODO APP to help users organize their plans efficiently. Built on the MERN (MongoDB, Express.js, React.js, Node.js) stack, the platform offers a range of features tailored for a seamless user experience and enhanced productivity.

Key Features:

One-Time Purchase Option: A hassle-free purchase option via PayPal or credit/debit cards is available, enabling users to unlock the task management features with a single transaction.

Task Management: Users can effortlessly create, edit, and delete tasks, streamlining their planning process.

Priority-Based Visuals: The app intuitively color-codes task cards based on their priority, providing a quick visual cue for urgent tasks.

Deadline Alerts: Tasks nearing their deadline within the next two days are highlighted with a gradient red background, ensuring timely attention.

Authentication System: Leveraging Firebase, the platform ensures a secure and straightforward account creation and login process.

Live Support Chat: A real-time chat feature connects users directly with the admin for instant support. While users can interact solely with the admin, the admin gains a comprehensive view of all ongoing chats.

Day and Night View: Users can switch between light and dark modes for enhanced visual comfort and usability.

Technologies Used:

Client-side:

React 18.2.0
Tailwind for UI styles
React Query for data fetching and state management
Ant Design for UI elements
Axios for HTTP requests
Firebase for authentication
Socket.io-client for real-time communication

Server-side:

Node.js with Express.js for backend development
MongoDB for database management
Socket.io for enabling real-time chat functionalities
CORS for handling cross-origin requests
dotenv for environment configuration

Development Process:

Planning and Initial Setup:
The development journey began with thorough planning to identify the core requirements and features of the todo list application. A clear roadmap was established to guide the MERN stack development process, outlining the user flow, essential functionalities, and desired user experience.

Frontend Development:
React was chosen for the frontend due to its component-based architecture. TailwindCSS handled UI styling, while Tanstack Query managed data fetching and state.

Backend Development:
Node.js with Express.js formed the backend, with MongoDB as the database. API endpoints managed task operations, and Socket.io enabled real-time chat.

Integration between Frontend and Backend:
The frontend-backend integration used RESTful API calls and Socket.io. Axios handled HTTP requests, linking the frontend to backend services.

Authentication Implementation:
Firebase provided a secure authentication system. It managed user registration and login processes, seamlessly integrated into the frontend.

Additional Features Implemented:
In addition to the core todo list functionalities, several enhanced features were implemented to enrich the user experience:

One-Time Purchase Option: A seamless purchase option via PayPal or credit/debit cards was integrated, enabling users to unlock premium features with a single transaction.

Live Support Chat: A real-time chat feature was implemented to connect users with the admin for instant support. While users could interact solely with the admin, the admin had a comprehensive view of all ongoing chats, facilitated by Socket.io for real-time communication.

Priority-Based Visuals: Task cards were color-coded (blue for low, yellow for moderate, red for high) based on their priority, providing a visual cue for urgent tasks.

Deadline Alerts: Tasks nearing their deadline within the next two days were highlighted with a gradient red background, ensuring timely attention.

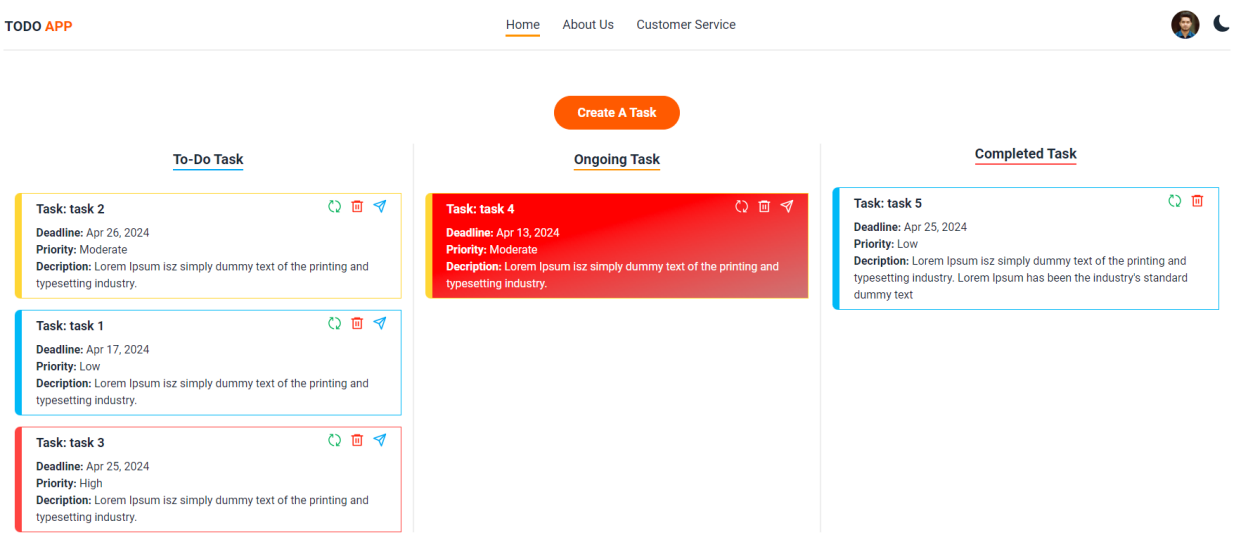

Figure: Priority visual & deadline alerts

Challenges Faced:

While building the todo list app, I ran into a couple of tricky spots. First, getting the drag and drop feature to work smoothly was tougher than I expected. After trying a few things, I found `react-beautiful-dnd`, which really helped sort that out. The other big challenge was setting up Socket.io for the live chat. That took some digging into the docs, blogs, and Stack Overflow to figure out. After spending some time researching and testing, I managed to get everything working. So, with a bit of persistence and help from the community, I was able to tackle those hurdles and make the app better.

Code Structure and Quality:

The code for the todo list app is well-organized, readable, and follows best practices. It's structured with separate frontend and backend directories and clear categorization of components, pages, hooks and others. Descriptive naming and consistent formatting enhance readability. The code adheres to industry standards, utilizing component-based architecture, Tanstack Query for the purpose of state management, comprehensive error handling, and security measures like Firebase authentication and HTTPS protocols.

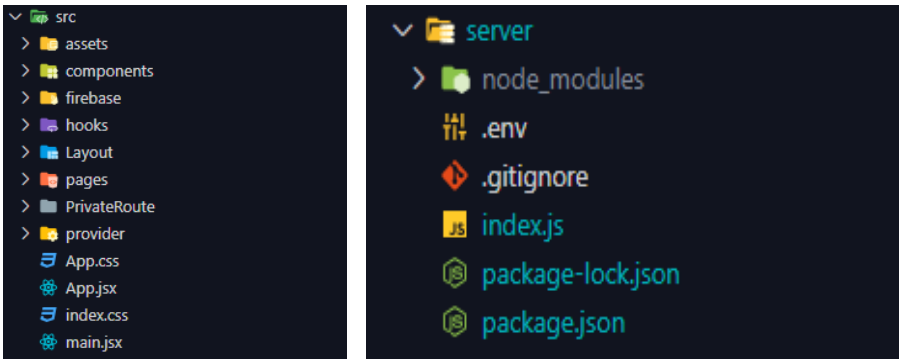

Figure: File structures of the client and server side

User Experience:

Responsiveness:
The app is optimized for all screen sizes, adapting layouts dynamically for desktops, tablets, and mobiles. This ensures consistent accessibility and usability across devices.

Intuitiveness:
The interface features a clean layout with clear navigation and interactive elements. The drag and drop feature allow easy task movement between 'todo', 'ongoing', and 'complete', simplifying task management. Dedicated buttons further streamline this process.

Feedback Mechanisms:
A live chat feature is integrated for real-time admin support and queries. This enhances user engagement by providing immediate and personalized assistance, building trust and fostering a positive user experience.

Deployment and Instructions:

Deployment Process:
The application is deployed on AWS due to the specific requirements of real-time data handling for Socket.io. While using the free version of AWS, occasional glitches may be experienced. But locally you will not face any issue.

AWS Setup Instruction:

1. **Sign up for a free AWS account at AWS.**
2. **Create and launch an EC2 Instance**
3. **SSH into Machine:** Open your terminal, navigate to the key folder, and paste the SSH key to access your EC2 instance.
4. **Update the package list and check if Git is installed:**
   - *sudo apt update*
   - *git –version*
5. **Install Node.js and NPM**
   - *curl -fsSL https://deb.nodesource.com/setup_21.x | sudo -E bash*
   - *sudo apt-get install -y nodejs*
6. **Clone Git Repository**
   - git clone [git url]
7. **Install npm**
   - Enter the git repository and install npm packages for the client and server directory
8. **Install PM2 globally and start your app:**
   - sudo npm i pm2 -g
9. **Start Server**
   - Navigate to ther server folder
   - Run *pm2 start index.js*

Local Setup Instructions:

Download Node (if you don't have it in your laptop):

- Go to https://nodejs.org/en/download
- Download the LTS version and install it in your laptop.

Server Setup:

- Open VS code and Navigate to the server directory in the terminal (for windows, ctrl + J).
- Write ***npm i*** to install necessary packages.
- Write ***npm run dev*** and server will start.

Client Setup:

- Open VS code and Navigate to the client directory in the terminal (for windows, ctrl + J).
- Write ***npm i*** in the terminal to install necessary packages.
- Write ***npm run dev*** and client will start.

Live view:

- Open http://localhost:5173 in your browser to view the running site.

Additional Comments:

Given more time, I would focus on:

UI Design: Refining the design for a more modern and user-friendly look.
Security: Implementing enhanced security measures and best practices.
Performance: Optimizing page loading times and caching for improved speed.
Post-Deployment: Resolving any AWS free tier limitations and addressing post-deployment issues for reliable operation.

Finally, I would like share my gratitude for giving me this opportunity. After the in-person interview where you shared your background, goals and current plan, I am very much excited than before to join your company. That is why I tried to give more than you asked for this job assignment. Actually, this is the way of my work ethics, try to do as much as I can add within the timeframe and go beyond the requirements.

I have necessary skills and I have the mind to learn, grow and adapt in any situation. I never feel boring or afraid to learn new technology, new things. I am a very much dedicated person. I don't care about 8 hours working per day, if I need to stay longer hours, I can do that where it is paid or not, doesn't matter to me. Even If you call me in the middle of a night for a work that company needs or someone to help with their workload, I would love to do that. Because I believe in, we win, I win.

I am a developer, that doesn't limit my work. I would love to help in other aspect of my office needs, like administration, accountings etc. if I have the knowledge and time after my work. I am friendly, organized and helpful minded person. That is all about my work ethics that I wanted to share. So, if you are hesitating between me and some other candidates, please give me a chance. I will make sure that I can deliver your needs.