**Thought Note**

# **Project Structure**

* + **Sales App**
    - **frontend**
      * *package.json*
      * node\_modules/
      * public/
        + index.html
        + …
      * src/
        + ***Componesnts***

NavBar.css

NavBar.js

* + - * + ***Pages***

AddSales.js

Form.css

Login.js

Register.js

TodayTotalRevenue.js

TopSales.js

* + - * + ***App***.js

…

* + - **backend**
      * public/
        + index.html
        + …
      * ***middleware***
        + protectedResource.js
      * ***models***
        + product\_model.js
        + user\_model.js
      * node\_modules/
      * ***routes***
        + product\_route.js
        + user\_route.js
      * ***config***.js
      * *package-locke.json*
      * *package.json*
      * ***server***.js

# **Modules Installed**

## **BACK END**

* + npm install **express**
  + npm install **mongoose**
  + npm install **cors**
  + npm install **jsonwebtoken**
  + npm install **bcrypt**

## **FRONT END**

* + npm install **axios**
  + npm install **sweetalert2**
  + npm install **react-router-dom**
  + npm install **redux react-redux**

# **Integration for Design**

To enhance the visual appeal and design consistency of the sales app two things were integrated into the project

1. **Bootstrap**
2. **Google Fonts**

It was implemented with the help of **CDN** instead of installing them as libraries.

# **Dynamic Routing**

Installed the package “**react-router-dom**” to enable dynamic routing with smooth navigation between different sections of the sales app.

# **Custom Styling for Improved UI**

Applied custom styling on the following

* **Hamburger** Icon button
* **Submit** Button of the Form
* Overall **margin** and **padding** of various sections in sales app.

# **Back end Summary**

The backend project adheres to a structured and scalable architecture, emphasizing modularity and best practices in Full Stack Web Development. The key components of the project structure are outlined below:

## **middleware**

### **protectedResource.js:**

Implements middleware responsible for handling protected resources, bolstering the overall security posture. This middleware is related to authentication of user and JSON web tokens. Basically, if implemented it wont allow any page if user is not logged in.

## **models**

### **product\_model.js:**

Defines the schema for product-related data, ensuring a consistent and organized representation. This mongoose schema for product ensures proper data type and required status for product name, quantity and amount as all three of them are required as far as validation and assignment is concerened.

### **user\_model.js:**

Establishes the data model for user-centric information, promoting a standardized approach to user data. This mongoose schema helps us in registration process making sure that user enters proper first name, last name, email and password as far as data type and mandatory fields.

## **routes**

### **product\_route.js:**

Arranges routes related to product functionality, encapsulating logic related to product operations. This file pertains to adding product, retrieving sales data, such as, top 5 sales data and total revenue.

### **user\_route.js:**

Manages routes associated with user-centric operations, capturing user-related functionality. This file pertains registration and login of user making sure that password is in hashed format making security as a primary concern with the help of bcrypt module.

## **config.js:**

Centralized configuration file covering settings and parameters critical for the application's functionality.

## **server.js:**

Core server file arranging the execution of the application, covering the primary server logic and initializing the backend components.