**THOUGHT NOTE**

**Backend**

The first thing I did while starting with the backend of the sales application was to create a folder structure

* I started with creating a file and named it server.js which served as the entry point of the backend application then created an “app.js” file to keep everything related to “express” separately. After that I create a “config.env” file to store the environment variables like – connection string, port information, etc
* I created a folder with the name “Router” to store all the route files for different purposes eg- userRouter, and addsaleRouter. After that created a Controller folder to store the route handler functions and finally created a Modal folder to store the Modal like user modal, sale modal

.

* As usual started by creating a package.json file. I installed various packages which were required for creating the backend like express, mongoose, cors, nodemon, jsonwebtoken, bcrypt, validator, dotenv etc
* Talking about the server.js file, first thing first started with importing the environment variables, and then establishing the connection with the MongoDB database with the help of “Mongoose”
* It should be noted that I used “Mongodb Atlas” which is a cloud database to store all the data instead of storing the data locally. (ensured that all the IP addresses can access the database by configuring the Atlas setting while creating the “Cluster”😅
* Talking about app.js file, I imported the express library and then used various “middlewares” to carry out different actions like app.use(cors()), to allow cross-origin
* Talking about config.env file, in this file I stored all the “connection string” of the database, DatabasePassword, username, JWT token expiration duration in seconds, port information on
* Talking about the Modals folder, after creating the connection with MongoDB database. I created two files, one for “userModal” and the other for “addSaleModal” , the idea was to create “Schema”, which is just like field that the application would contain, and which will be provided by the user. For talking about “userModal” , passed in the fields like firstname, lastname, email, password, confirmpassword.
* On the Schema I set the “validations”, used custom validations as well as imported a library “Validator.js” to perform the validation.Once the Schema was made, I created a “modal” out of it on which I performed various Operations like creating, getting a data etc.. in the route handler function.
* Talking particularly about userModal, here I used a “preSave hook” of mongoose , to encrypt the password before storing it to the database for that I took the help of “bcrypt” library and also created the “instance” method to compare the password while checking that the user entered the correct password or not
* Talking about Router folder, in here I created two separate files one for “useRouter” and the other for “addSaleRouter”,
* Talking about Controller folder, in here I created two files, one with the name of authController and the other with the name AddSaleController. These controller file consisted of the “route handler function” to carry to operation.
* In the “authController.js” file I created handlers for “signup”, “login” and “protected route”, used the “JSONwebtoken” for the authentication work,(by installing the JSONwebtoken library)
* Talking the “addSaleController.js” file , I created handler for “adding new sale”, “gettingtop5sales”, “total revenue for today” for that I used – “aggregation” pipeline. In order to calculate the total revenue for today, I used match stage and group stage, took the product of quantity and price to get the amount and the adding the amount to get the total revenue

**-----------------------------------------------------------------------------**

**Frontend**

**First of all, I created a folder with the name “Sales-Frontend”. Then I installed React from**

**npm using the command npx create-react-app and gave it the name “salesfrontend”.**

* **As per the requirement of the project. I created two folders inside the “src” folder with the name – components and pages respectively.**
* **I created a Redux folder to store the state of the user for that I installed the “redux and react-redux” library**
* **Inside the components folder, I created various components like Navbar.js , Revenue.js, etc**
* **The idea was to create a navbar that would contain routes to all the other pages like total revenue, add sales, login, logout etc**
* **I took the help of Bootstrap to design the “navbar” and all the other pages**
* **Added “Google fonts” library in the “index.html” file which is the “Public folder”**
* **I used axios library to send the http request**
* **I used the “sweetalert2” package for sending customized alert to the client**
* **Made the design responsive as well using Bootstrap classes**
* **Took the help of “react-router-dom” package to enable the implementation of routing**

**# Talking separately about different pages I designed-**

* **Talking about the redux folder, I created two folders 1) Action and 2) Reducer, and a file “store.js”**

**Inside the “Action” folder I created a “userAction.js” to define the two actions namely , “loginSucces” and “loginFailure”, then in the “Reducer” folder I created two files one, userReducer.js where I called the userReducer function to act and then exported this to the “rootReducer” file**

**And then finally created a “store.js” file to create a centralized store and then passed the state to the root component app.js) with the help of the “provider” component of react-redux**

1. **Add Sales – simply use the “form” class of Bootstrap to design the layout and ensure that the page is responsive. Here the user will fill in the data to add the sales, the user will provide the -Product name, its quantity, and the “price of one item”.**
2. **Top 5 Sales - since this page would display the top 5 sales that were made today, so I used the “table” class of Bootstrap to design the layout and ensure that the page is responsive.**
3. **Today’s Total Revenue - This is a simple page that would simply display the total revenue generated today, so for that, I used the “h1” tag to give it the desired font size and also applied some Google font to it to make it look appealing**
4. **Login and Registration – to design this I simply used the “form” class of Bootstrap to design the layout and ensured that the page was responsive. Used the JWT for authentication. Stored the token in the local storage of the browser and utilized the token as a “pass” to carry out other actions on the app, like adding sales, getting top sales or revenue**
5. **Logout – clicking on this the user will logout from the application**

* **It should be noted that I rendered the nav items conditionally using short-circuiting using the “and” operator. when you start the application you will only be able to view two options, “login” and “registration”. Once you are successfully registered and logged in to the app, only then you will be able to – addSales, get revenue or top 5 sales.**

**------------------------------------------------------------ The End---------------------------------------------------------**