Module 5 – Sales App with MERN

(Putting it all together – Creating Web-App Part 2)

- By Ajeet Kumar (Batch: 15th June)

Project Title: Sales Tracker

Date Prepared: 12th November 2023

Developed By: Ajeet Kumar

GitHub Repository: https://github.com/ajeetkumarrauniyar/Sales-App-UI

Introduction

The **Sales Tracker** is a comprehensive and dynamic full-stack web application, meticulously crafted using the MERN (MongoDB, Express.js, React, Node.js) stack and featuring React.js for its versatile and interactive user interfaces. This application is designed to efficiently manage sales data, user authentication, and deliver essential insights, including top daily sales and total revenue.

Project Structure

The system follows a client-server architecture each having its own package.json and node modules. This separation allows for clear organization of the frontend and backend code.

- **Client Side:** The client side consists of web browsers through which users interact with the system's graphical user interface (GUI).

- **Server Side:** The server side is built using Node.js and Express.js. It handles incoming HTTP requests, processes data, and communicates with the storage system.

Resources Used:

- Visual Studio Code as the code editor
- Google Chrome as the web browser
- POSTMAN to test the APIs
- MongoDB Compass for the Database
- Bootstrap 5 as the frontend framework imported via CDN.
- Google Fonts “Poppins” as custom font, imported via CDN.

Technology Stack

- **Frontend:** React
- **Backend:** Node.js, Express.js
- **Database:** MongoDB
- **Middleware:** JWT for authentication
- **Libraries:** Mongoose for MongoDB interactions, Bcrypt for password hashing, react-hot-toast for client-side notifications.

Application Highlights:

1. **Navigation Excellence:** Seamless navigation bar for easy access to key features.
2. **Streamlined Sales Entry:** User-friendly interface for adding sales entries.
3. **Top Sales Visualization:** Visual presentation of top 5 sales entries.
4. **Revenue Insight:** Clear view of today's total revenue in a concise card format.

User Authentication: Login and registration for authorized access to sales data.

Key Features:

1. **User Authentication:** The application ensures secure user registration and login using bcrypt for password hashing and JWT for authentication.
2. **Sales Tracking:** Allows the addition of sales entries with product details, calculates total amounts, and stores the data in MongoDB.
3. **Data Insights:** Provides endpoints for retrieving top sales of the day and calculating total revenue.
4. **Middleware Security:** Implements middleware to protect certain routes (e.g. Add Sales) by validating JWT tokens.

Getting Started / Project Setup

To get started with the Sales Tracker, follow these steps:

1. Clone the repository to your local machine.

   git clone https://github.com/ajeetkumarrauniyar/Sales-App-UI

2. Navigate to the project directory.

   cd client ,
   cd server

3. Install the required dependencies by running the following command:

   npm install

4. Start the development server.

   npm start

Server-Side Components

- **Server Entry Point (server.js)**

  The **server.js** file is the entry point for the server. It sets up the Express application, connects to the database, includes models, routes, and starts the server on the specified port.

- **Environment Variables (.env)**
  The .env file contains configuration variables such as the server port, database URL, and JWT secret.

- **Database Configuration (config/dbConfig.js)**

  The **dbConfig.js** file is responsible for connecting the application to the MongoDB database. It uses the Mongoose library to establish a connection. If the **DATABASE_URL** is not specified in the environment variables, it defaults to a local MongoDB instance. The connection status and the connected database's name are logged for verification.

- **Sales Model (models/salesModel.js) and User Model (models/userModels.js)**

  The **salesModel.js** and **userModels.js** files define Mongoose schemas for sales entries and user data, respectively. These schemas are used to create and register models that interact with the MongoDB database.

- **Authentication Routes (routes/authRoutes.js)**
  The **authRoutes.js** file defines routes for user registration and login. These routes use the controllers from authController.js.

- **Authentication Controllers (controllers/authController.js)**

  The **authController.js** file contains controllers for user registration (registerUser) and login (loginUser). Key functionalities include:

  - **Registration:** Validates mandatory fields, checks for existing users with the same email, hashes the password, and saves the new user to the database.

  - **Login:** Validates credentials, checks for the user's existence, compares passwords, and generates a JWT token for authentication.

- **Authentication Middleware (middleware/authMiddleware.js)**
  The **authMiddleware.js** file defines middleware for protecting resources by verifying JWT tokens. It extracts the token from the request headers, verifies it, and attaches the user object to the request for subsequent middleware or routes.

- **Sales Routes (routes/salesRoutes.js)**
  The **salesRoutes.js** file defines routes for adding sales, retrieving top sales, and calculating total revenue. Middleware **(authMiddleware)** is used to protect the add sales route.

- **Sales Controllers (controllers/salesControllers.js)**
  The **salesControllers.js** file manages sales-related operations, including adding a new sale entry (addSalesEntry), retrieving the top 5 sales of the day (getTopSales), and calculating the total revenue (getTotalRevenue).

  - **Add Sales Entry:** Validates input, calculates the total amount, creates a new sale entry, and saves it to the database.
  - **Top 5 Sales of the Day:** Retrieves and aggregates data to find the top 5 sales of the current day.
  - **Total Revenue:** Calculates the total revenue by aggregating the amount field from all sales entries.

Future Improvements

1. **Error Handling:** Implement comprehensive error handling on both the client and server sides to provide meaningful feedback to users.

2. **User Authentication:** Enhance user authentication with features such as password reset functionality and user profile management.

3. **Frontend Enhancement:** Improve the user interface with additional features like data visualization, user-friendly forms, and interactive charts.

4. **Documentation:** Enhance code documentation to facilitate future development and collaboration.

Conclusion

The Sales Tracker App, developed using the MERN stack, is a robust full-stack web application featuring user authentication, data storage, and insights generation. With a focus on best practices in authentication and data handling, the app establishes a strong foundation for future enhancements.

Built with React.js, the Sales Tracker offers an intuitive interface for efficient sales data management, including features like adding sales, viewing top sales, and tracking daily revenue. The MERN stack implementation ensures a modular structure, security measures, and effective use of MongoDB, contributing to a scalable and efficient application.

This project highlights React.js's capabilities in creating modern and responsive user interfaces. With planned improvements, the Sales Tracker App has the potential to be a powerful tool for businesses in managing and analyzing sales data, providing a solid foundation for further development based on specific business requirements.