Module 4 – **Sales App UI Design with React.js**

(Putting it all together – Creating Web-App Part 1)

**- By Ajeet Kumar (Batch: 15th June)**

**Project Title: Sales App UI Design**

**Date Prepared: 30th September 2023**

**Developed By: Ajeet Kumar**

**GitHub Repository:** <https://github.com/ajeetkumarrauniyar/Student-management-system>

# **## Introduction**

The **Sales App UI** is a dynamic web application meticulously crafted using React.js, a versatile JavaScript library renowned for its capacity to build interactive and responsive user interfaces. This project report delves into the comprehensive development process, highlights the key features and functionalities of the Sales App UI, and outlines the project's architecture and setup.

# **## System Architecture**

The Sales App UI follows a single-page application (SPA) architecture using React.js. It is a client-side application that provides a user-friendly interface for managing sales-related data. The primary components of the system include:

* **App.js:** This serves as the main entry point of the application and handles routing to various pages.
* **HeaderComponent.jsx:** This component provides navigation links and a search bar for user interaction.
* **AddSales.jsx**: Empowers users to effortlessly add sales entries by offering product selection, quantity, and amount input fields.
* **TopSales.jsx:** Showcases the top-performing five sales entries in an aesthetically pleasing and responsive table format.
* **Revenue.jsx:** Provides an at-a-glance view of the day's total revenue in an elegantly designed card.
* **Login.jsx:** Renders a login form for user authentication.
* **Register.jsx:** Extends the user experience with a registration form for new users.

**## Resources Used:**

* Visual Studio Code as the code editor
* Google Chrome as the web browser

**## Key Features:**

## **Navigation Excellence**

The application boasts an exceptional navigation bar, enabling users to seamlessly switch between various sections of the app. It offers easy access to functionalities such as adding sales, viewing top sales, checking daily revenue, and user authentication.

## **Streamlined Sales Entry**

The Sales App UI provides a user-friendly interface for adding sales entries. Users can add sales entries by selecting a product, entering the quantity, and the amount. This data is then stored for analysis and reporting.

## **Top Sales Visualization**

The **"Top Sales"** section is a visual delight, presenting the top 5 sales entries in a table format. The presentation includes product names, quantities, and sale amounts, allowing users to gain valuable insights effortlessly.

## **Revenue Insight**

In the **"Revenue"** section, users are presented with a succinct view of today's total revenue. A clear and concise card format, complete with a currency symbol (₹), facilitates quick tracking of daily sales performance.

## **User Authentication**

The application includes a user authentication system with login and registration pages, allowing authorized users to access and manage sales data.

**## Getting Started / Project Setup**

To get started with the Sales App UI, follow these steps:

1. Clone the repository to your local machine.

**git clone** [GitHub Repository URL]

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

**npm install**

1. Start the server:

**node index.js**

# **## Implementation**

The system's implementation involves key components and libraries:

* **Express.js:** Express is used to create the web server and handle HTTP requests.
* **Body Parser:** The Body Parser middleware is used to parse request data, enabling the system to handle POST requests effectively.
* **Node-Persist:** Node-Persist is used as a storage mechanism for persistently storing student data. It allows data to be saved and retrieved between server sessions.

# **## Features, Code Overview & Usage**

The core functionality of the system is implemented in the **`index.js`** file. Here is a summary of the key routes and functions:

## **Welcome Screen**

**Code Overview**

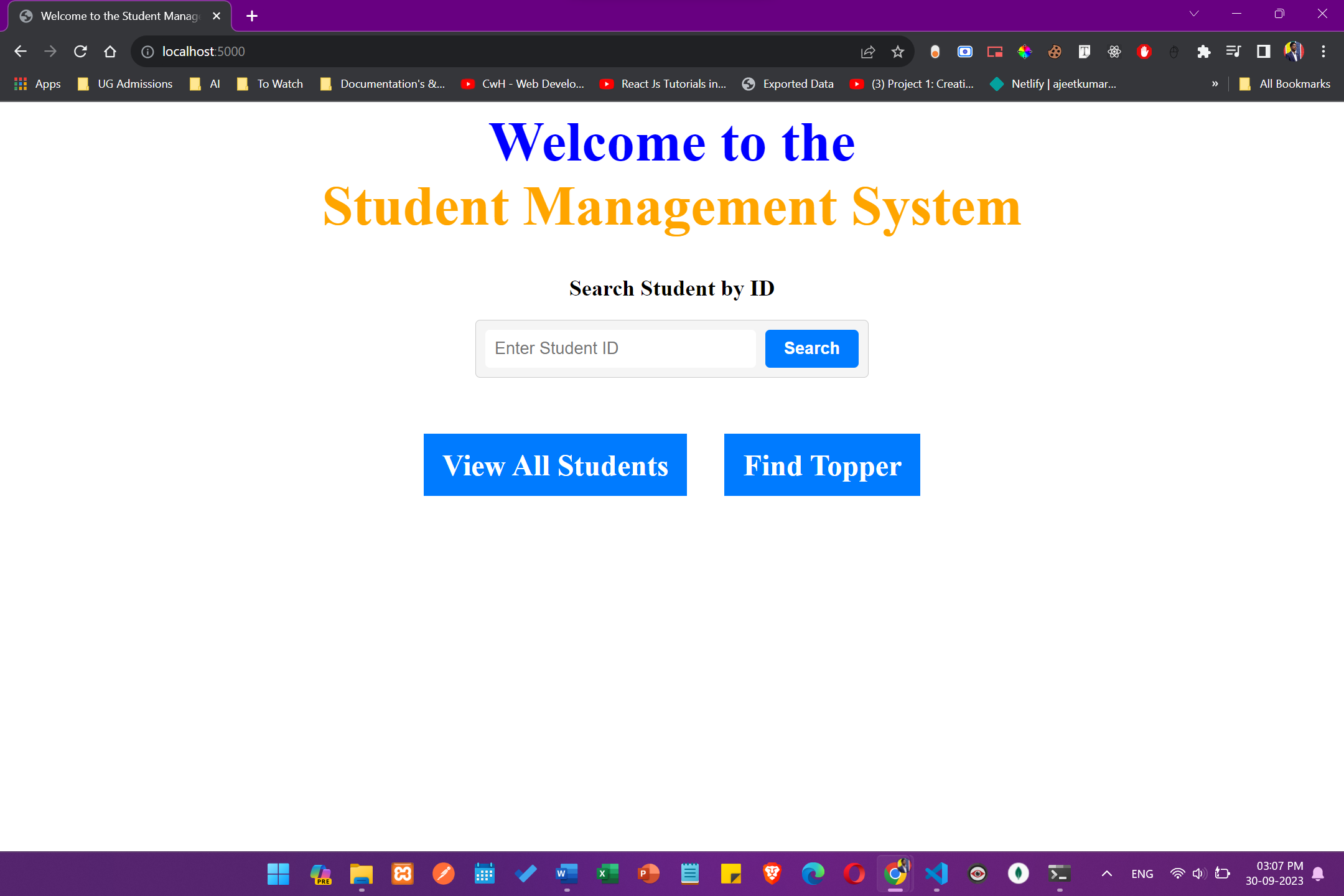
* The default route displays a welcome screen with options to search for students, view all students, and find the topper.

**Features**

* Upon accessing the system, users are greeted with a visually appealing welcome screen.
* Users can:
  + Enter a student ID to search for a specific student.
  + View a list of all students.
  + Find the top-performing student based on GPA.

**Usage**

* Access the system's welcome screen by navigating to [**http://localhost:5000**](http://localhost:5000)in your web browser. You will be greeted with a welcome message and several options.



## **Adding Student Data**

**Code Overview**

* POST requests to **`/student`** are used to add student data to the storage. Data is extracted from the request body and stored.

**Features**

* The system allows the addition of student data. Users can provide the following information:
  + Student ID
  + Student Name
  + GPA
* Upon submission, the student data is stored in the Node-persist system.

**Usage**

* To add a new student to the system, make a POST request to **`/student`** using a tool like **POSTMAN** with the following JSON payload:

```json

{

"student\_id": "your\_student\_id",

"student\_name": "student\_name\_here",

"gpa": "student\_gpa\_here"

}

```

A screenshot of a computer

Description automatically generated­­­­

## **Retrieving (or Viewing )All Student Data**

**Code Overview**

* A GET request to **`/allStudents`** retrieves and formats all student data for display.

**Features**

* Users can access a dedicated page to view a list of all students. The system retrieves student data from storage and presents it in a structured manner.

**Usage**

* To view a list of all students stored in the system, access [**http://localhost:5000/allStudents**](http://localhost:5000/allStudents). You will see a formatted list of student data.

A screenshot of a computer

Description automatically generated

## **Retrieving Student Data by ID**

**Code Overview**

* A GET request to **`/student/:id`** allows searching for students by ID. It displays student data if found, or an error message if not found.

**Features**

* Users can search for a specific student by entering their ID. If the student exists in the system, the system displays their data. Otherwise, an error message is shown.

**Usage**

* To view the details of a specific student by their ID, access [**http://localhost:5000/student/:id**](http://localhost:5000/student/:id), replacing **`:id`** with the actual student ID.

A screenshot of a computer

Description automatically generated

**5. Finding the Top-Performing Student**

**Code Overview**

* A GET request to **`/topper`** identifies the student with the highest GPA and displays their data.

**Features**

* The system identifies the top-performing student based on GPA and displays their data. This feature is helpful in recognizing high-achieving students.

**Usage**

* To find the top-performing student based on GPA, access [http://localhost:5000/topper. You will be presented with information about the student with the highest GPA.

A screenshot of a computer

Description automatically generated

**6. Searching for Students by ID**

* POST requests to **`/searchStudent`** search for students by ID and display the result or an error message.

A screenshot of a computer

Description automatically generated

**## Conclusion**

The Student Management System provides an efficient way to manage student data for educational institutions. Its user-friendly interface, along with features such as adding, searching, and displaying student information, makes it a valuable tool for administrators and educators.

Future enhancements could include user authentication, data validation, and the ability to update student records. Overall, the system serves as a solid foundation for managing student data and can be further extended to meet specific institutional requirements.

This project demonstrates the use of Node.js and Express.js to create a functional web application for managing data, which can be applied to various domains beyond education.