Module 1 – Node & Express

(Developing the backend functionality using Node & Express)

- By Ajeet Kumar (Batch: 15th June)

Project Title: Student Management System

Date Prepared: 29th September 2023

Deployed On: https://github.com/ajeetkumarrauniyar/Student-management-system

Introduction

The **Student Management System** is a web application developed using **Node.js** and **Express.js.** It provides features such as adding, searching, and displaying student data. This report details the development process, features, and usage of the system.

System Architecture

The system follows a client-server architecture:

- **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. Storage system is built using the **Node-persist.**

Resources Used:

- Visual Studio Code as the code editor
- Google Chrome as the web browser
- POSTMAN to add the data.

Getting Started

To get started with the Student Management System, follow these steps:

1. Clone the repository to your local machine.
   https://github.com/ajeetkumarrauniyar/Student-management-system

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

   npm install

3. 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:

1. 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

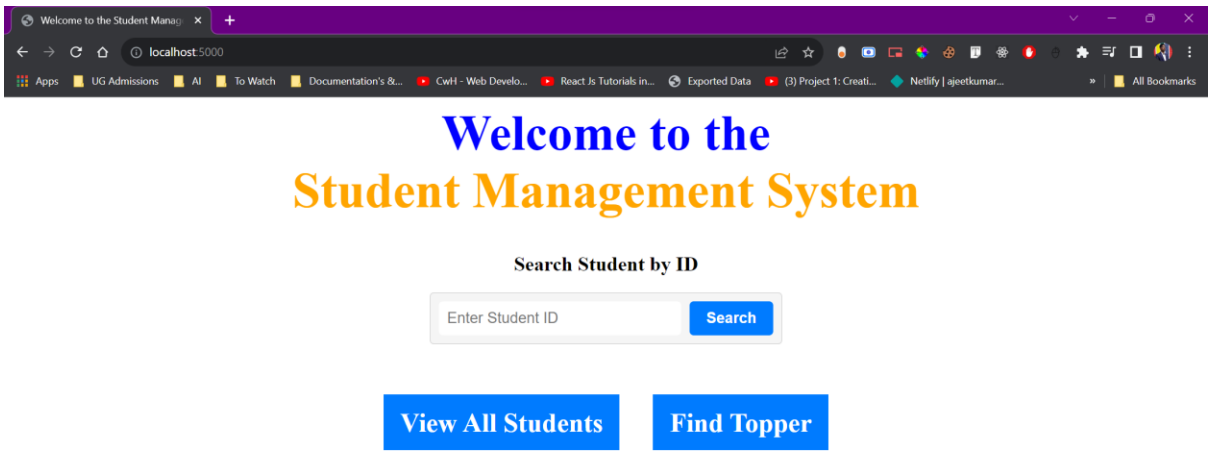
- Upon accessing the system, users are greeted with a visually appealing welcome screen.

- 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 in your web browser. You will be greeted with a welcome message and several options.



2. 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.

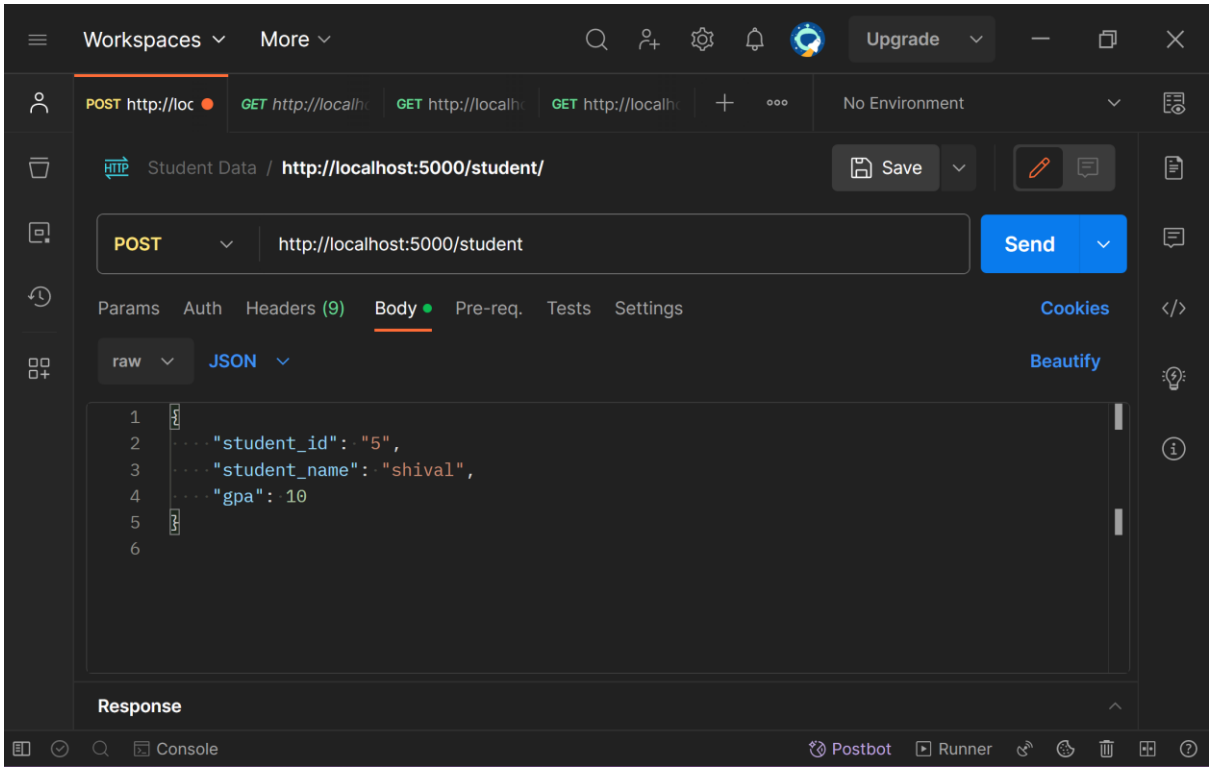
Features

- 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"
}
```



3. Retrieving (or Viewing)All Student Data

Code Overview

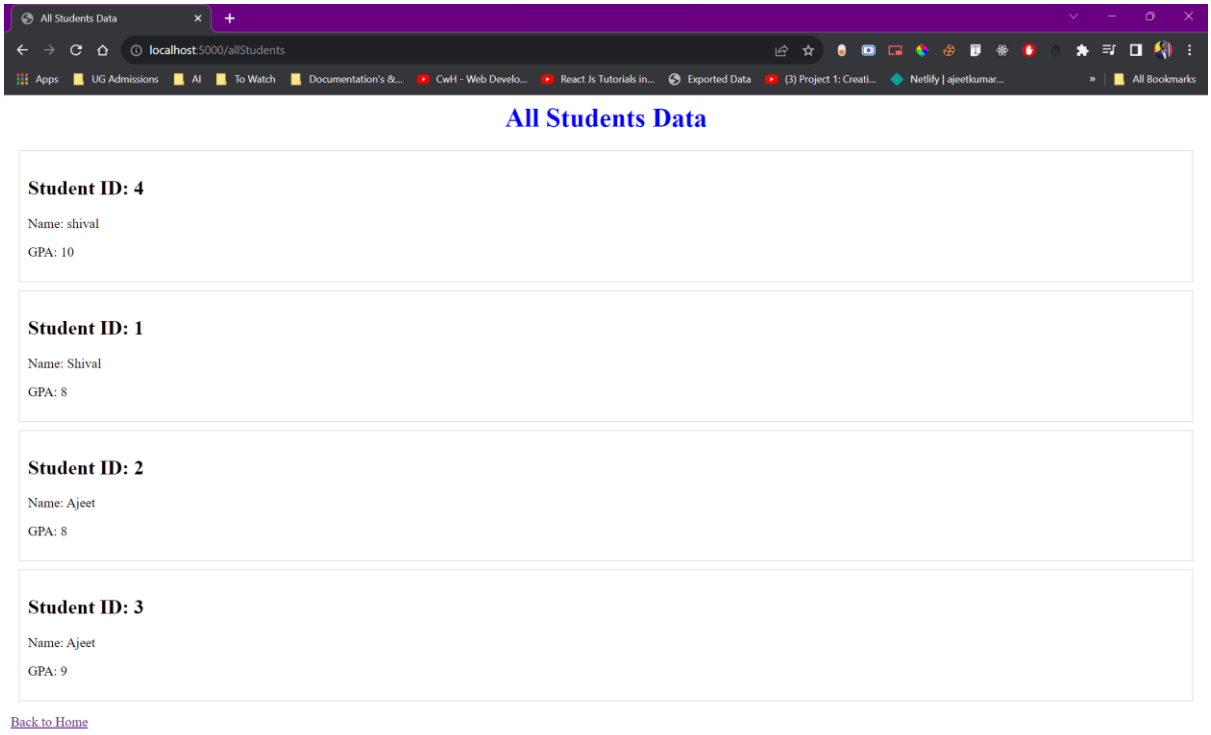
- A GET request to `**/allStudents**` retrieves and formats all student data for display.

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** . You will see a formatted list of student data.



---

4. Retrieving Student Data by ID

Code Overview

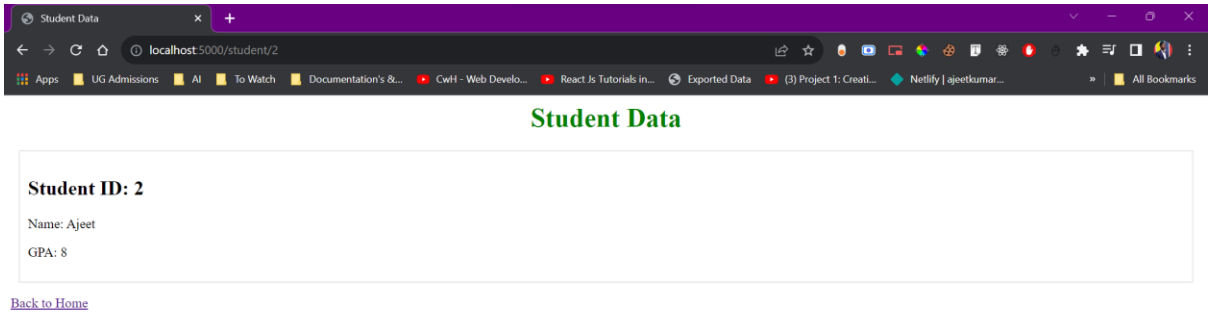
- A GET request to `**/student/:id**` allows searching for students by ID. It displays student data if found, or an error message if not found.

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** , replacing `**:id**` with the actual student ID.



5. Finding the Top-Performing Student

Code Overview

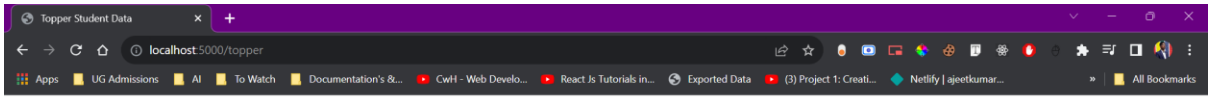
- A GET request to `**/topper**` identifies the student with the highest GPA and displays their data.

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.



Topper Student Data

Student ID: 4

Name: shival

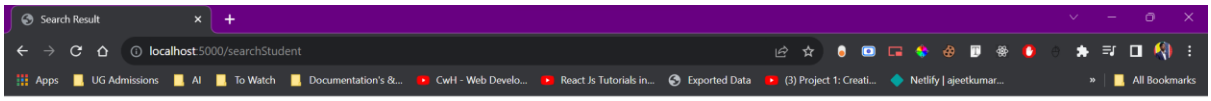
GPA: 10

Back to Home

---

6. Searching for Students by ID

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



Search Result

Student ID: 2

Name: Ajeet

GPA: 8

Back to Home

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.