MUHAMMAD TAYYAB
21JZBCS0157

Lab Report: Node.js CRUD Application with Bash Scripting (JQuery)

Abstract:

This lab report documents the development of a Node.js application that implements basic CRUD (Create, Read, Update, Delete) operations for managing student data. The application utilizes bash scripts to interact with specific API endpoints exposed by the Node.js server.

Introduction:

This project aimed to create a simple Node.js application capable of managing student information. The application provides functionalities to create new student entries, retrieve all students or a specific student based on ID, update existing student information, and delete students. Bash scripts act as user interfaces for interacting with the application's functionalities.

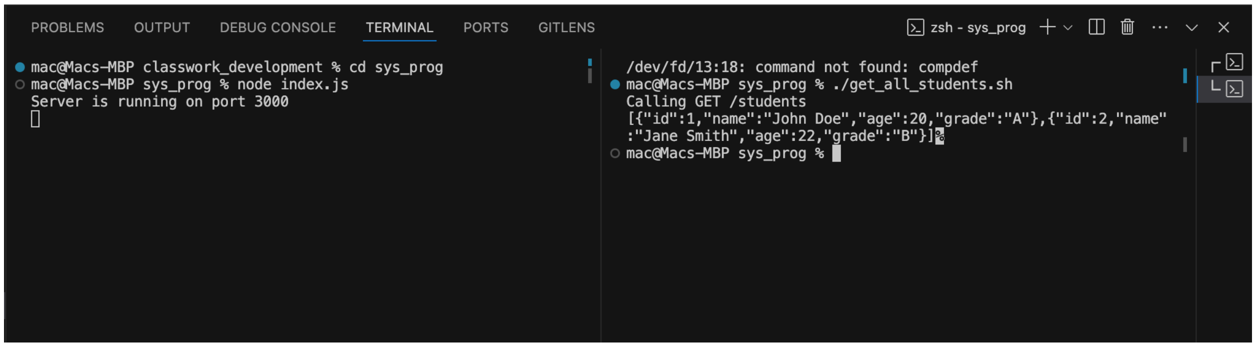
System Design:

The system consists of two main components:

- **Node.js Server:** This server application handles all data processing and manipulation. It exposes various API endpoints for CRUD operations on student data.
- **Bash Scripting:** A set of bash scripts serve as user interfaces for invoking specific API endpoints on the Node.js server. These scripts leverage jQuery (potentially) to handle user input and make HTTP requests.

Functionalities:

1. **Get All Students (/students):**
    - This functionality retrieves a list of all students stored in the database.
    - **Image:**

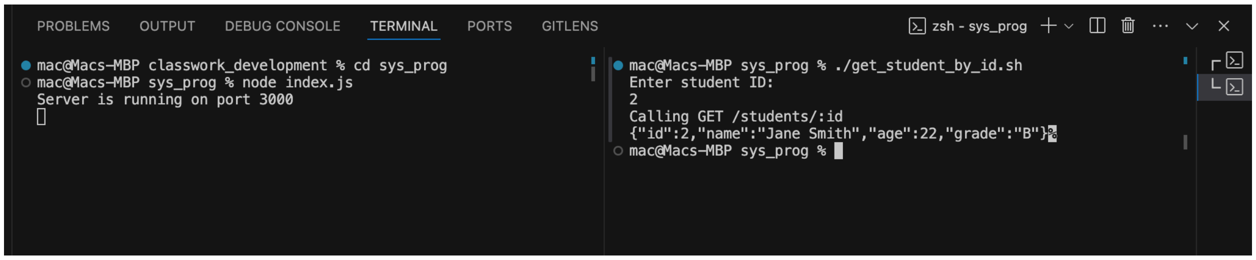


```
mac@Macs-MBP classwork_development % cd sys_prog
mac@Macs-MBP sys_prog % node index.js
Server is running on port 3000
```

```
/dev/fd/13:18: command not found: compdef
mac@Macs-MBP sys_prog % ./get_all_students.sh
Calling GET /students
[{"id":1,"name":"John Doe","age":20,"grade":"A"},{"id":2,"name"
:"Jane Smith","age":22,"grade":"B"}]%
mac@Macs-MBP sys_prog %
```

    - **Description:** The user executes the ./get_all_students.sh script. The Node.js server logs messages indicating it's running and performs a GET request to /students. The server responds with JSON data containing an array of student objects, each with properties like "id", "name", "age", and "grade".

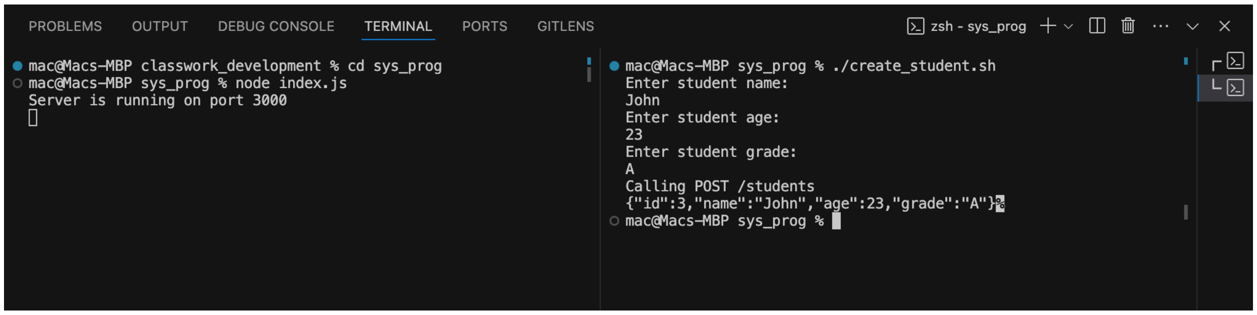
2. **Get Student by ID (/students/:id):**
   - This functionality retrieves information for a specific student based on their ID.
   - **Image:**



```
mac@Macs-MBP classwork_development % cd sys_prog
mac@Macs-MBP sys_prog % node index.js
Server is running on port 3000
```

```
mac@Macs-MBP sys_prog % ./get_student_by_id.sh
Enter student ID:
2
Calling GET /students/:id
{"id":2,"name":"Jane Smith","age":22,"grade":"B"}%
mac@Macs-MBP sys_prog %
```

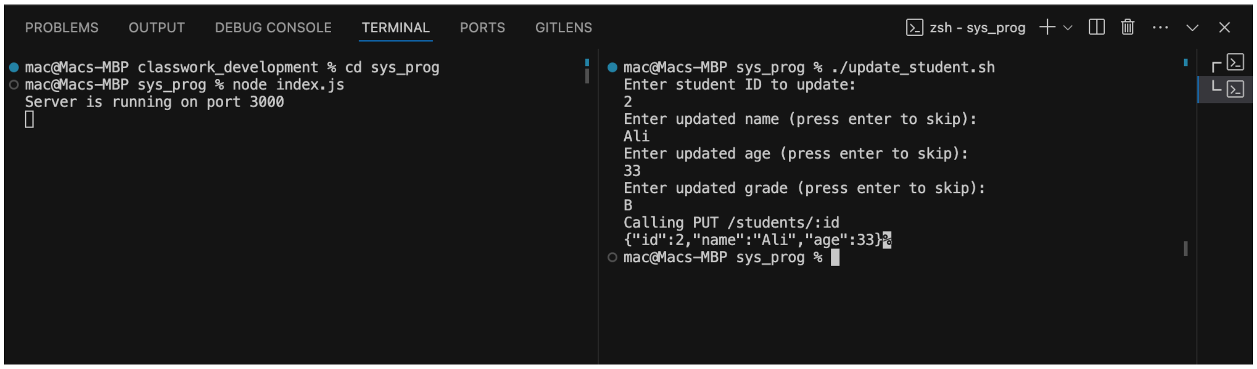
   - **Description:** The user executes the node index.js script. The server logs messages and performs a GET request to /students/:id, where ":id" is replaced with the actual student ID. The response includes a JSON object containing detailed information about the retrieved student.
3. **Create Student (POST /students):**
   - This functionality allows adding a new student to the system.
   - **Image:**



```
mac@Macs-MBP classwork_development % cd sys_prog
mac@Macs-MBP sys_prog % node index.js
Server is running on port 3000
```

```
mac@Macs-MBP sys_prog % ./create_student.sh
Enter student name:
John
Enter student age:
23
Enter student grade:
A
Calling POST /students
{"id":3,"name":"John","age":23,"grade":"A"}%
mac@Macs-MBP sys_prog %
```

   - **Description:** The user executes the ./create_student.sh script. The script prompts the user to enter details like name, age, and grade. This information is then sent to the server via a POST request to /students. The server processes the data, creates a new student record, and returns a JSON object with the newly created student's information.
4. **Update Student (PUT /students/:id):**
   - This functionality allows modifying information for an existing student.
   - **Image:**



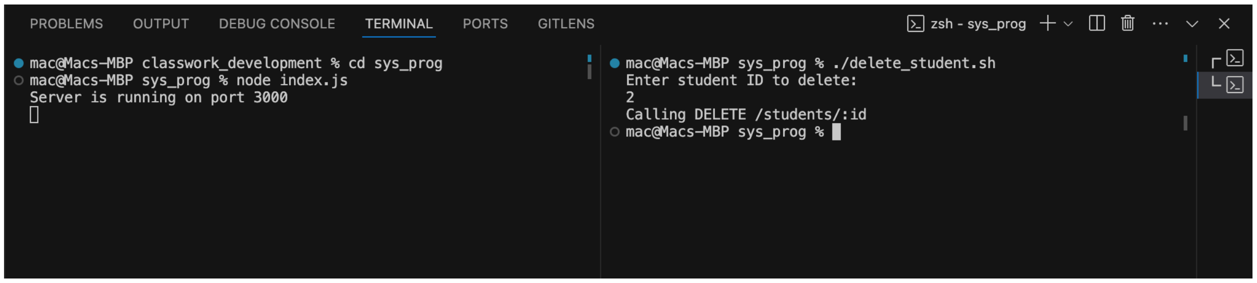
```
mac@Macs-MBP classwork_development % cd sys_prog
mac@Macs-MBP sys_prog % node index.js
Server is running on port 3000
```

```
mac@Macs-MBP sys_prog % ./update_student.sh
Enter student ID to update:
2
Enter updated name (press enter to skip):
Ali
Enter updated age (press enter to skip):
33
Enter updated grade (press enter to skip):
B
Calling PUT /students/:id
{"id":2,"name":"Ali","age":33}%
mac@Macs-MBP sys_prog %
```

   - **Description:** The user executes the node index.js script. The script prompts the user to enter the ID of the student to be updated. It then allows updating specific fields like name, age, and grade. The script sends a PUT request to /students/:id with the updated information. The server processes the request, updates the student record, and returns a

JSON object reflecting the changes.

5. **Server:**
   - **Image:**



```
mac@Macs-MBP classwork_development % cd sys_prog
mac@Macs-MBP sys_prog % node index.js
Server is running on port 3000
```

```
mac@Macs-MBP sys_prog % ./delete_student.sh
Enter student ID to delete:
2
Calling DELETE /students/:id
mac@Macs-MBP sys_prog %
```

   - **Description:** This image shows the server running on port 3000. The specific details of other functionalities are obscured by the process list and other windows.

Conclusion:

This lab report presented a Node.js application with bash scripting for CRUD operations on student data. The application demonstrates the basic functionalities of creating, retrieving, updating, and deleting data using a server-client architecture. Further development could involve implementing functionalities like user authentication, data validation, and error handling.