**Module 3: Lab 2 Custom Modules**

Saurabh Kale

IFT 458/554: Middleware Programming & Database Security

Dinesh Sthapit

Sep 24th, 2023

Part 1:

Server started-

A computer screen shot of a program

Description automatically generated

Output-

A screenshot of a computer

Description automatically generated

Added few records-

A screenshot of a computer

Description automatically generated

In student.js:

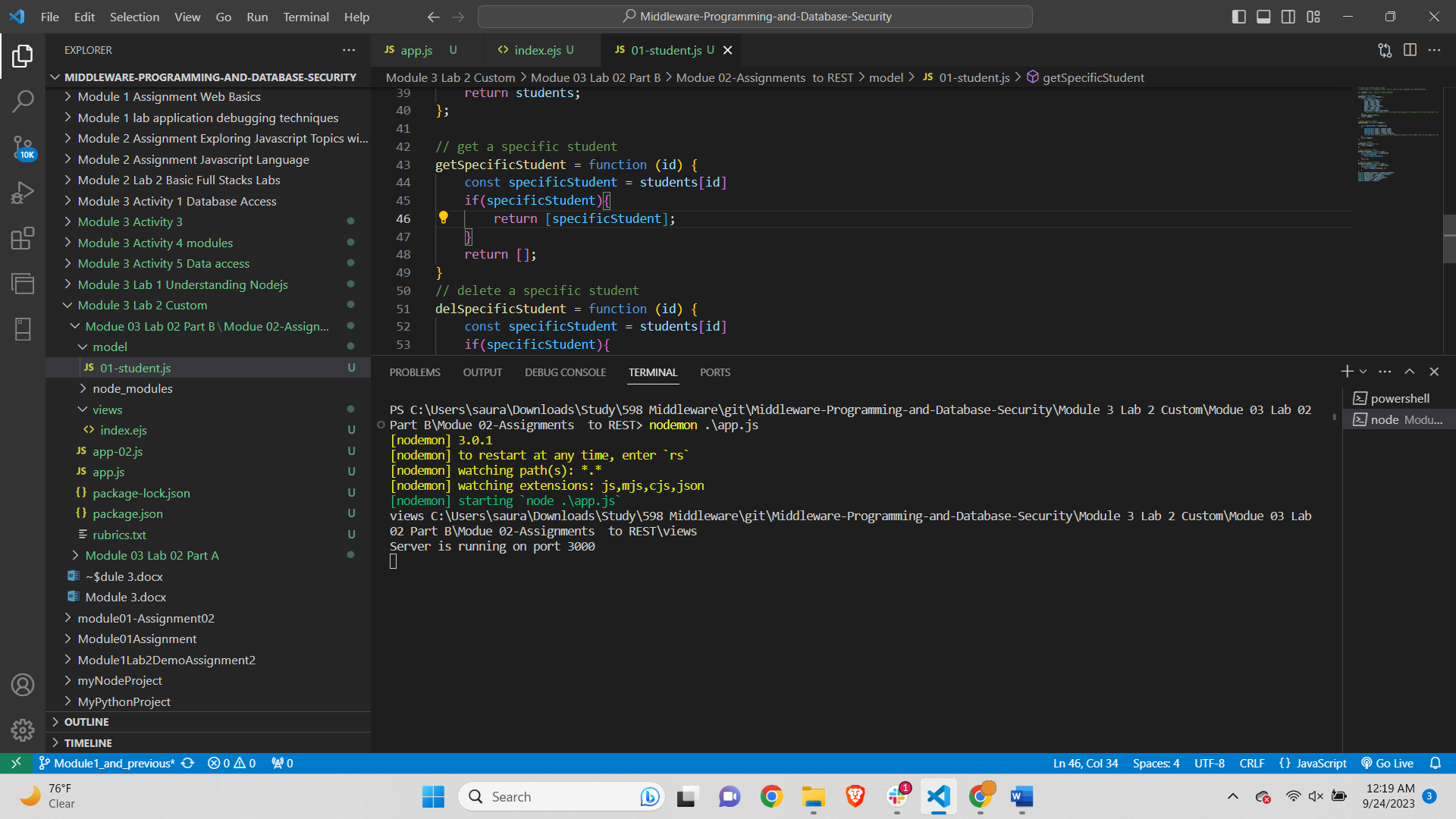
* addStudent, updateStudent, and getStudents functions each have a single responsibility related to managing student data. addStudent adds a new student to the list, updateStudent updates a student's information, and getStudents retrieves the list of students. These functions are well-focused on their respective tasks and do not attempt to do more than they should. Therefore, student.js follows the SRP.

In index.js:

* This file is primarily an HTML template for rendering student information and a form for adding new student data. It doesn't have much logic but is responsible for displaying data to the user. So, it follows the SRP for this purpose.

Part 2-

Server started-



Form-

A screenshot of a computer

Description automatically generated

Displaying 2 entries-

A screenshot of a computer

Description automatically generated

First entry in the students array-

A screenshot of a computer

Description automatically generated

Second entry in the array-

A screenshot of a computer

Description automatically generated

In comparing the code you provided in Part 1 with the code in your follow-up message, there are several key differences and observations:

1. Additional Functionality:
   * The code in Part 1 focused on creating a basic student information form, allowing users to input data and display it. It did not implement CRUD (Create, Read, Update, Delete) operations.
   * The code in your follow-up message has been significantly expanded to include CRUD operations. It now allows for adding, updating, retrieving specific students, and deleting students from a data store.
2. Express.js Usage:
   * In Part 1, you were setting up a basic Express.js server to render an HTML form and display student data, but it didn't include routes for CRUD operations.
   * In your follow-up code, you have extended the Express.js application to handle various HTTP routes (GET, POST, PUT, DELETE) for managing student data, which adds complexity to the application.
3. Separation of Concerns:
   * In your follow-up code, there is a clear separation of concerns between routes and the data store (implemented in students.js). This separation aligns with the Single Responsibility Principle (SRP) by isolating the handling of data operations in one module.
4. EJS Templates:
   * Both versions of the code use EJS templates to render HTML pages. However, your follow-up code has added new HTML forms and templates for CRUD operations, making it more sophisticated and comprehensive in terms of user interaction.
5. Overall Complexity:
   * The code in your follow-up message is more complex and sophisticated compared to the code in Part 1. This is primarily due to the addition of CRUD functionality and the associated routes. The code now handles a wider range of tasks related to managing student data.

**POSTMAN-**

**GET-**

A screenshot of a computer

Description automatically generated

**POST-**

**A screenshot of a computer

Description automatically generated**

**PUT request-**

**A screenshot of a computer

Description automatically generated**

**DELETE request-**

**A screenshot of a computer

Description automatically generated**