CN5006 WEEK-04

Outputs for the exercises are attached below:

**Exercise-01 Basic Express Application**

This exercise involved setting up the core Express server and defining the initial “Hello” route on Port 5000.

Terminal code:

A screenshot of a computer

AI-generated content may be incorrect.

Browser Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise-02 Adding Routes and URL Parameters**

Exercise 2 extended the application to include two additional routes, demonstrating the use of static and dynamic URL parameters.

Browser Output:

* For **/about**:

A screenshot of a computer

AI-generated content may be incorrect.

* For **/users/33/books/123:**

**A screenshot of a computer

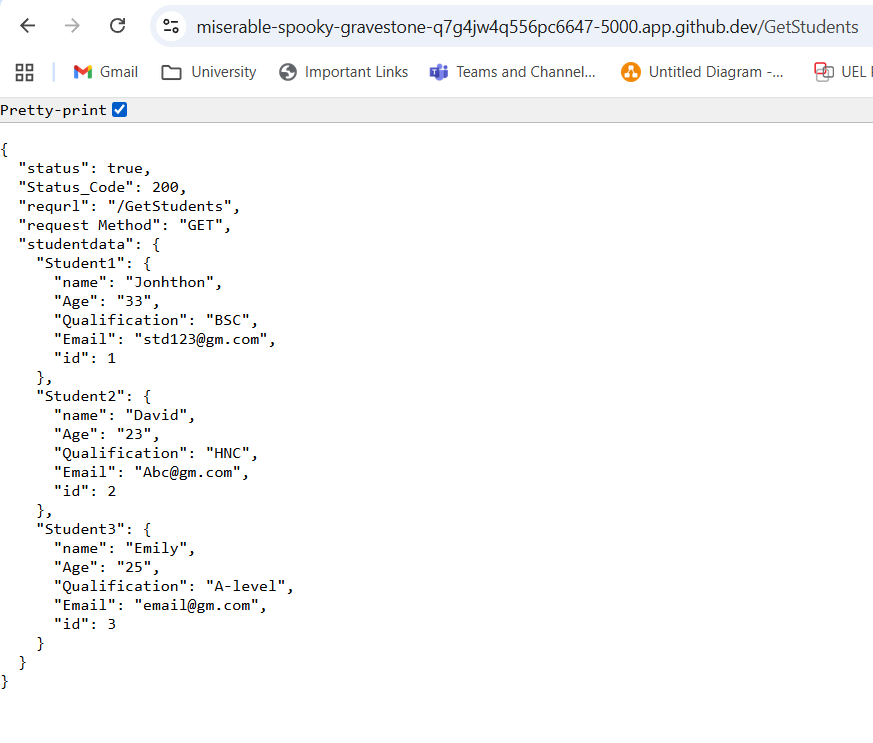
AI-generated content may be incorrect.**

**Exercise-03 Reading JSON data**

This exercise integrated file system operation to read student data from student.json and serve it via two GET endpoints, one for all records and one for the specific ID.

Browser Output:

* For all records:



* For specific IDs:

1. GetStudentid/1

A screenshot of a computer

AI-generated content may be incorrect.

1. GetStudentid/2

A screenshot of a computer

AI-generated content may be incorrect.

1. GetStudentid/3

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise-04 Implementing the POST Method**

This final exercise involved creating an HTML form and back-end POST route that used body-parser to manage the incoming data and return a structured JSON confirmation.

Browser Output:

* Without filling up the information

A screenshot of a computer

AI-generated content may be incorrect.

* After the information is filled:

A screenshot of a computer

AI-generated content may be incorrect.

* After hitting the submit button:

A screenshot of a computer

AI-generated content may be incorrect.

**Reflective Report:**

A great hands-on introduction to creating a RESTful service with Node.js and the Express framework was given in Week 4 lab. The learning objectives were successfully met, especially when it came to handling various HTTP methods and dynamic routing.  
  
**Important Takeaways:**

* Express Initialization: I became adept at configuring a simple Express server and specifying the required middleware, like body-parser, which is essential for managing incoming data from client-side forms.
* Routing and parameters: I had a clear understanding of how to define different routes (/, /about, and parameterized routes like /users/:id/books/:bookid) and how to use the req.params object to access dynamic portions of a URL. This is crucial for developing scalable and adaptable APIs.
* Data Handling (Files and JSON): Exercise 3 was especially instructive since it showed how to read and parse local JSON files using the fs module that comes with Node.js. Combining this with URL parameters allowed for the creation of simple data retrieval endpoints (/GetStudent/:id).
* HTTP Techniques: The lab highlighted the function of the req.body object in the latter by distinguishing between the implementation of a simple GET request (for data retrieval) and a POST request (for data submission).

**Obstacles and Overcoming Them:**

Making sure the body-parser middleware for Exercise 4 was integrated correctly was the biggest obstacle faced. At first, the POST route's req.body object would return undefined. This was fixed by carefully making sure everything was set up correctly, including the body-parser import and using app.use(bodyParser.urlencoded({ extended: true })); before the POST route definition to properly parse the form data sent by the HTML file.

To sum up, this lab strengthened my grasp of the Express.js framework and its function in creating contemporary REST APIs by offering a solid, practical foundation for creating back-end services.