Syllabus: Backend using Node.js & SQL

Unit 1: Introduction to Node.js and Express.js

- Overview of Node.js and NPM(Node Package Manager)
- Introduction to Express.js and its features
- Setting up a basic Express.js server

Unit 2: Building API Routes

- Introduction to RESTful API design principles
- Creating API routes with Express.js to handle different HTTP methods
- Responding with JSON data from the database
- Using Postman for Testing API endpoints and validating responses

Unit 3: MySQL Basics

- Introduction to relational databases and MySQL
- Understanding tables, columns, and relationships
- Basic SQL queries for CRUD operations

Unit 4: Connecting Node.js with MySQL

- Establishing a connection to MySQL database using Node.js
- Executing MySQL queries with the mysql2 library
- Handling asynchronous operations with Promises

Unit 5: Authentication and Authorization

- Understanding authentication and authorization concepts
- Implementing user authentication with Express.js
- Using session-based and token-based authentication
- Implementing password hashing

Unit 6: API security

- Implementing middleware for request processing
- Implementing error-handling middleware in Express.js
- Handling common security issues (SQL injection, XSS attacks, etc.)
- Logging errors and application events for debugging and monitoring

Unit 7: Deployment and API Documentation

- Deploying the API to a hosting platform
- Generating API documentation using tools like Swagger

Tasks

▼ **Task 1(unit 2):** Create a calculator API

- Calculator API that takes three inputs as per below
- Design an API that takes three inputs: `number1`, `number2`, and `operation`.
- Supported operations are: addition (`add`), subtraction (`sub`), multiplication (`mul`), division (`div`), and modulo (`mod`).
- output response: The API should respond with a JSON object containing the input values and the result of the operation.

```
{
  data: {
    number1: 10,
    number2: 11,
    operation: "add",
    result: 21,
  },
}
```

- Bonus:
  - Implement validation for the inputs:
    - Check if all three values are provided.

        - Validate that the operation is one of the supported operations.
    - Use appropriate HTTP response codes such as 200 for success and 400 for bad requests.

▼ **Task 2(Unit 3):** CRUD operations on Employees' table

create a table named `Employees` with the following columns:

- `id` (integer, primary key)
- `name` (string)
- `department` (string)
- `salary` (decimal)

Write SQL queries to perform the following CRUD operations on the `Employees` table:

1. Create:
    - Write an SQL query to insert 10 new employee with the following details
2. Read:
    - Write SQL queries to retrieve
        - all employees from the table.
        - the employee with the highest salary
        - all employees who have a salary more than the average salary.
3. Update: Update the salary of the employee with the highest salary to 6000.00.
4. Delete: Delete an employee with a specific name from the table.

▼ **Task 3(Unit)4:** API for performing CRUD operations on Employees' table

creating an API to perform CRUD operations on an "Employees" table in a MySQL database. The table has the following columns:

- `id` (integer, primary key)
- `name` (string)
- `department` (string)

- `salary` (decimal)

Using Node.js and Express.js, design and implement an API with the following endpoints:

1. GET `/employees`: Retrieves all employees from the database.
2. GET `/employees/:id`: Retrieves a specific employee by their ID.
3. POST `/employees`: Creates a new employee in the database. Accepts a JSON payload with the employee details (name, department, and salary).
4. PUT `/employees/:id`: Updates an existing employee's information. Accepts a JSON payload with the updated employee details.
5. DELETE `/employees/:id`: Deletes an employee from the database based on their ID.