Practical No. 7

Study and implementation of Express.js

Perform following problem statements using ReactJs

Problem Statement 1: Basics of Express.js

- What is Express.js and how does it differ from Node.js?
- How do you create a simple Express.js server?
- Explain the concept of routing in Express.js. How do you define routes?
- What is middleware in Express.js, and how does it work?
- How do you create and use custom middleware in an Express.js application?
- What is the difference between application-level middleware and router-level middleware?
- What are req and res in Express.js? Give examples of common properties and methods associated with each.
- How would you extract query parameters from a URL in an Express.js route?
- How does Express.js handle different HTTP methods (GET, POST, PUT, DELETE)?
- What are route parameters in Express.js? How do you use them in a route definition?

Problem Statement 2: Basic Web Server with Express.js

Requirements

- Create a basic Express.js server that listens on port 3000.
- Define three routes:
    - GET / - Responds with "Welcome to the Home Page".
    - GET /about - Responds with "This is the About Page".
    - GET /contact - Responds with "Contact us at: email@example.com".
- Include a 404 error handler that displays a "Page Not Found" message for unknown routes.

Problem Statement 3: Dynamic Route Parameters

Requirements

- Modify the previous server to include the following route:
- GET /users/:id - Responds with "User ID: [id]" where [id] is the dynamic value from the route.
- Add another route:
    - GET /products/:category/:productId - Responds with "Category: [category], Product ID: [productId]".

- Return a JSON object containing the category and product ID instead of a plain string.Note:

1. Create a **document** of the above website with screenshots.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.