**Batch: T5**

**Practical No. 7**

**Title of Assignment:Study and implementation of Express.js**

**Student Name: Trishul Balaso Patil**

**Student PRN:22510051**

**Perform following problem statements using ReactJs**

**Problem Statement 1: Basics of Express.js**

**1.What is Express.js and how does it differ from Node.js?**

Node.js is a runtime environment that allows you to run JavaScript on the server, providing tools for building basic servers, handling files, and performing networking tasks.

Express.js is a web framework built on top of Node.js that simplifies web server creation by adding features like routing, middleware support, and HTTP utility methods, making it easier to build APIs and web applications.

**2)How do you create a simple Express.js server?**

Ans.

const express = require('express');

const app = express();

app.listen(3000, () => {

console.log('Server listening on port 3000');

});

**3)Explain the concept of routing in Express.js. How do you define routes?**

Ans. Routing in Express.js determines how requests are handled based on the URL. You define routes using the app. METHOD(path, handler) syntax, where METHOD is the HTTP method (e.g., GET, POST), path is the URL pattern, and handler is a callback function that handles the request.

**4)What is middleware in Express.js, and how does it work?**

Ans:Middleware functions are invoked for every request and response. They have access to the request and response objects, as well as the next middleware function in the chain. Middleware can perform tasks like logging, authentication, and error handling.

**5)How do you create and use custom middleware in an Express.js application?**

Ans:You can create custom middleware functions by defining a function that takes req, res, and next as arguments.

**6)What is the difference between application-level middleware and router-level middleware?**

Ans:Application-level middleware applies to all routes in the application. Router-level middleware applies only to routes defined within a specific router instance.

**7)What are req and res in Express.js? Give examples of common properties and methods associated with each.**

Ans:req (request object) contains information about the incoming request, such as:

req.method: HTTP method (GET, POST, etc.)

req.url: URL path

req.query: Query parameters

req.body: Request body (for POST requests)

req.params: Route parameters

res (response object) is used to send a response to the client:

res.send(data): Sends a text or JSON response

res.json(data): Sends a JSON response

res.status(code): Sets the response status code

res.render(view, data): Renders a template

**8)How would you extract query parameters from a URL in an Express.js route?**

Ans:You can access query parameters using the req.query object.

**9)How does Express.js handle different HTTP methods (GET, POST, PUT, DELETE)?**

Ans: Express.js provides specific methods for each HTTP method:

**10)What are route parameters in Express.js? How do you use them in a route definition?**

Ans:Route parameters are placeholders in a URL that can be used to match different values

**Problem Statement 2: Basic Web Server with Express.js**

**Requirements**

**Create a basic Express.js server that listens on port 3000.**

**Define three routes:**

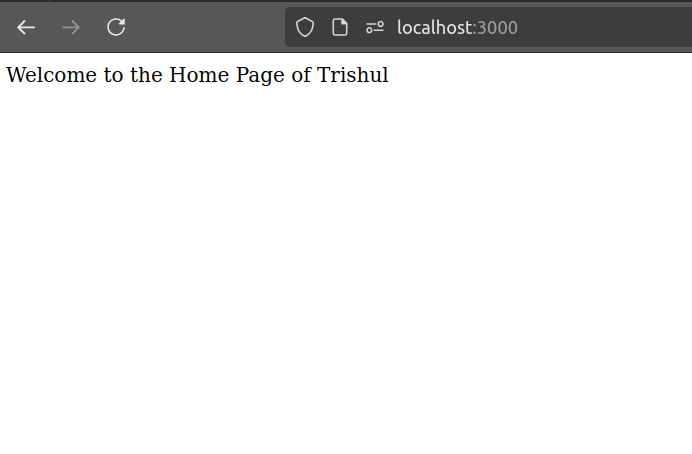
**o GET / - Responds with "Welcome to the Home Page".**

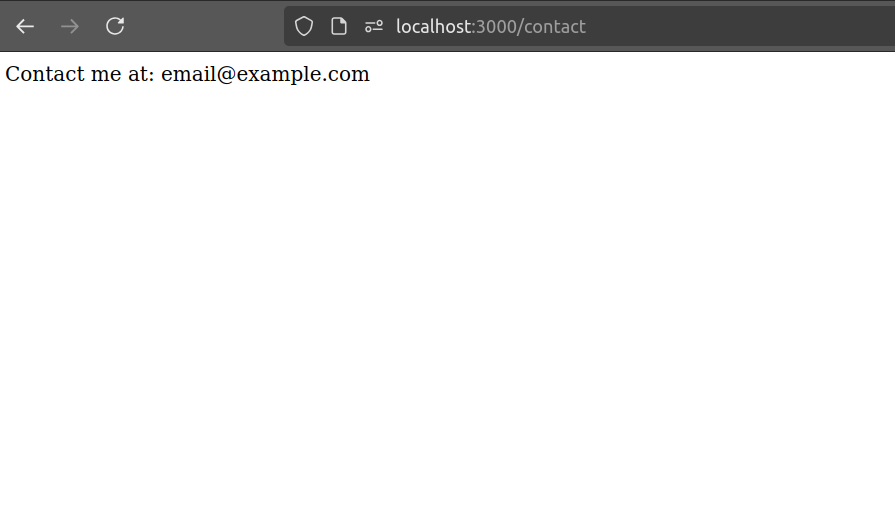
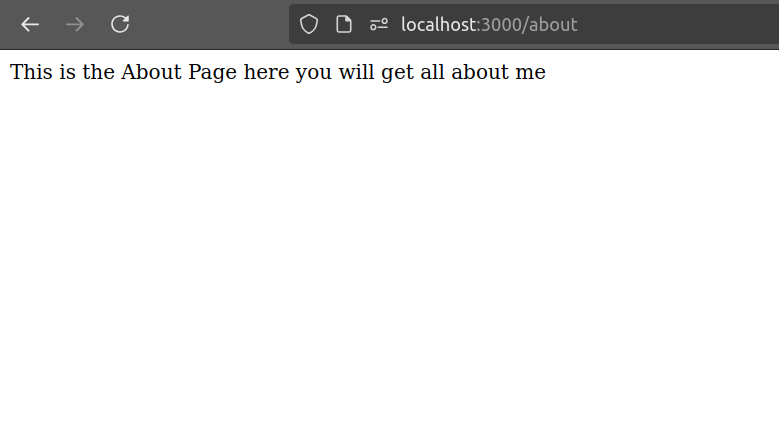
**o GET /about - Responds with "This is the About Page".**

**o 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.**

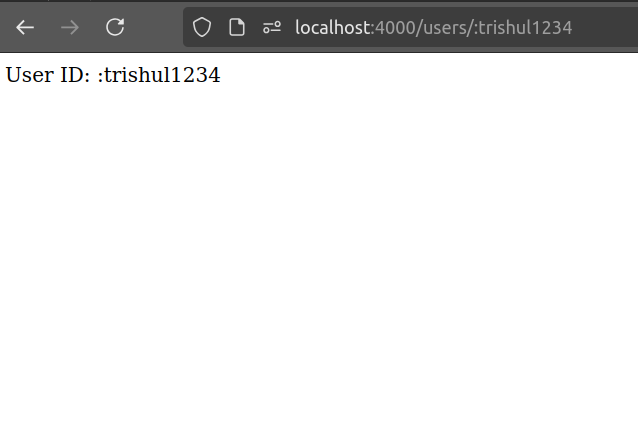
**Add another route:**

**o 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:**

****

****