**Name - Gaurang A Raorane Roll No - 49**

**Class - D15A Batch - C**

**EXPERIMENT NO - 10**

**Aim:-** Experiment to study basics of Node.js and Express.

**Theory:-**

Introduction to Node.js:

* + Node.js is an open-source, server-side runtime environment that allows you to run JavaScript outside of a web browser.
  + It is built on the V8 JavaScript engine and is known for its non-blocking, event-driven architecture.
* Setting Up Node.js:
  + Begin by installing Node.js on your computer. Node Package Manager (npm) is bundled with Node.js and will be essential for managing packages and dependencies.
* Understanding Modules:
  + Node.js uses a module system to organize code. You can create your own modules and use built-in modules.
  + Explore how to require and use modules in your applications.
* Creating a Basic Node.js Application:
  + Write a simple "Hello, World!" application in Node.js to understand the basic structure of a Node.js script.
* Asynchronous Programming:
  + Node.js is particularly powerful for handling asynchronous operations. Learn about callback functions and how to work with asynchronous code.
* Introduction to Express.js:
  + Express.js is a minimal, fast, and flexible Node.js web application framework that simplifies building web applications and APIs.
  + Understand the role of Express.js in the context of Node.js applications.
* Setting Up an Express.js Application:
  + Create a basic Express.js application, configure routes, and set up the server to listen for incoming requests.
* Routing and Middleware:
  + Explore routing in Express.js, which allows you to define how your application responds to different HTTP requests.
  + Learn about middleware and how it can be used to perform tasks such as authentication and request processing.
* Handling Requests and Responses:
  + Understand how to handle incoming HTTP requests, process data, and send appropriate responses.
* Template Engines and Views (Optional):
  + Learn about integrating template engines like EJS or Pug to generate dynamic HTML content.
* Database Connectivity (Optional):
  + If desired, delve into connecting your Express.js application to a database system like MongoDB or MySQL.

**Code:**

const http = require('http');

const net = require('net');

const fs = require('fs');

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello,World! This is the root route.');

});

app.get('/file', (req, res) => {

const filePath = 'example.txt';

const readStream = fs.createReadStream(filePath);

readStream.pipe(res);

});

const server = net.createServer((socket) => {

socket.write('Echo server\r\n');

socket.pipe(socket);

});

server.listen(1337, '127.0.0.1', () => {

console.log('Server listening on port 1337');

});

const httpServer = http.createServer((req, res) => {

res.writeHead(200, {'Content-Type': 'text/plain'});

res.end('Hello, World! This is an HTTP server.');});

httpServer.listen(8080, () => {

console.log('HTTP Server listening on port 8080');

});

app.listen(port, () => {

console.log(`Express server is running on port ${port}`);

});

net.createServer((socket) => {

socket.write('Echo server\r\n');

socket.pipe(socket);

socket.on('error', (err) => {

console.error('Socket Error:', err.message);

});

socket.on('close', () => {

console.log('Socket closed');

});

});

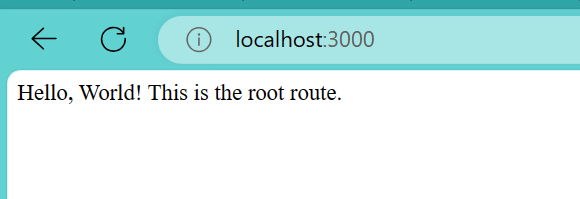
server.listen(1337, '127.0.0.1', () => {

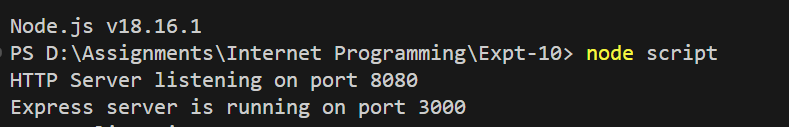
console.log('Server listening on port 1337');

});

**Output:**

**Fetched from example.txt**

****

****

**Program: express.js**

const express = require('express');

const app = express();

const port = 3000;

app.use(express.json());

app.get('/api/greet', (req, res) => {

const name = req.query.name || 'Guest';

res.send(`Hello, ${name}!`);

});

app.get('/api/user/:id', (req, res) => {

const userId = req.params.id;

res.send(`User ID: ${userId}`);

});

app.post('/api/user', (req, res) => {

const { name, email } = req.body;

res.status(201).json({ message: 'User created successfully', name, email });

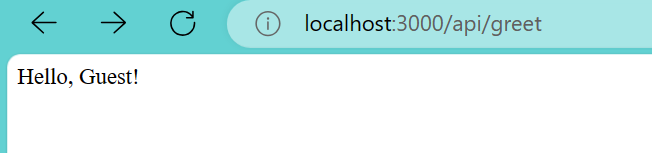
});

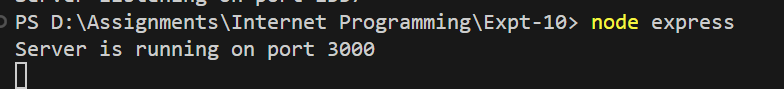
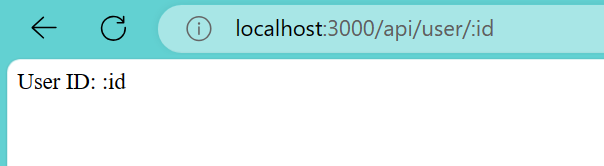
app.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

**Output:**

****

****