**AWT Lab Projects**

1. Create a Node Js server that listens to port 6001.
2. const http = require('http');
3. const server = http.createServer((req, res) => {
4. res.writeHead(200, { 'Content-Type': 'text/plain' });
5. res.end('Hello World!');
6. });
8. server.listen(6001, () => {
9. console.log('Server running on port 6001');
10. });

2.Implement URL parameter routing to display specific content based on the URL.(Eg: ‘/’, ‘/home’, ‘/profile’, etc.,)

// 2. Implement URL parameter routing to display specific content based on the URL.(Eg: ‘/’, ‘/home’, ‘/profile’, etc.,)

const express = require('express');

const app = express();

// Create a route for the root path

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

  res.send('Hello World!');

});

// Create a route for the /home path

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

  res.send('Welcome to the home page!');

});

// Create a route for the /profile path

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

  res.send('This is your profile page!');

});

// Start the server on port 6001

app.listen(6001, () => {

  console.log('Server listening on port 6001');

});

3. Create a route that returns a JSON response containing your name and email.

var express = require('express');

var bodyParser = require('body-parser');

var app = express();

//Note that in version 4 of express, express.bodyParser() was

//deprecated in favor of a separate 'body-parser' module.

app.use(bodyParser.urlencoded({ extended: true }));

//app.use(express.bodyParser());

app.post('/myaction', function(req, res) {

  res.json({'Name':req.body.name,'Email':req.body.email});

});

app.listen(6001, function() {

  console.log('Server running at http://127.0.0.1:6001/');

});

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

    <meta charset="utf-8">

    <title>Calculator</title>

</head>

<body>

    <!-- <div id="contact">

        <h1>Send an email</h1>

        <form action="http://127.0.0.1:6001/myaction" method="post">

            <fieldset>

                <label for="name">Name:</label>

                <input type="text" id="name" name="name" placeholder="Enter your full name" />

                <label for="email">Email:</label>

                <input type="email" id="email"  name= "email" placeholder="Enter your email address" />

                <label for="message">Message:</label>

                <textarea id="message" placeholder="What's on your mind?"></textarea>

                <input type="submit" value="Send message" />

            </fieldset>

        </form>

    </div> -->

</body>

</html>

4. Implement a route that accepts POST requests and logs the request data to the console.

// 4. Implement a route that accepts POST requests and logs the request data to the console.

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

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

    res.sendFile(\_\_dirname + '/index.html');

  });

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

 console.log('Name:', req.body.name);

 console.log('Email:', req.body.email);

 res.send('Form submitted successfully!');

});

app.listen(3000, () => {

 console.log('Server is running on port 3000');

});

Index.html

<form action="/submit" method="post">

            <label for="name">Name:</label>

            <input type="text" id="name" name="name" required><br><br>

            <label for="email">Email:</label>

            <input type="email" id="email" name="email" required><br><br>

            <input type="submit" value="Submit">

        </form>

5. Set up a static file server to serve HTML, CSS, and JavaScript files

var express = require('express');

var app = express();

var path = require('path');

app.use(express.static(path.join(\_\_dirname, 'public')));

app.get('/', function(req, res) {

    res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

app.listen(3000, function() {

    console.log('App listening on port 3000!');

});

6. Use the Express.js framework to create a basic web application with routing.

// 6.   Use the Express.js framework to create a basic web application with routing.

const express = require('express');

const app = express();

const port = 3000;

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

 res.send('Hello World!');

});

app.listen(port, () => {

 console.log(`Example app listening at http://localhost:${port}`);

});

7. Create a custom 404 error page for handling undefined routes.

// 7.   Create a custom 404 error page for handling undefined routes.

const express = require('express');

const path = require('path');

const app = express();

app.use(express.static(path.join(\_\_dirname, 'public')));

// Define all your other routes here...

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

    res.send('Hello world');

})

// Custom 404 error page middleware

app.use(function(req, res, next) {

 res.status(404).sendFile(path.join(\_\_dirname, 'public', 'index.html'))

});

app.listen(3000, function() {

 console.log('Server is running on port 3000');

});

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>Hello World</h1>

</body>

</html>

Index.html should be placed inside the public folder

8. Secure your routes with basic authentication using middleware.

const express = require('express');

const auth = require('express-basic-auth');

const authenticate = auth({

    users: { 'user': 'password' },

    challenge: true,

    realm: 'Private Area'

   });

   const app = express();

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

 res.send('Access granted to the secure route');

});

app.listen(3000, () => {

 console.log('Server is running on port 3000');

});

9. Create a RESTful API that returns data in JSON format.

// 9. Create a RESTful API that returns data in JSON format.

const express = require('express');

const mysql = require('mysql');

const app = express();

// Set up MySQL connection

const db = mysql.createConnection({

 host: 'localhost',

 user: 'root',

 password: '',

 database: 'student'

});

db.connect((err) => {

 if (err) throw err;

 console.log('Connected to the database');

});

// Define the GET /api/users route

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

 db.query('SELECT \* FROM users', (err, rows) => {

    if (err) throw err;

    console.log('Data received from the database:');

    console.log(rows);

    res.json(rows);

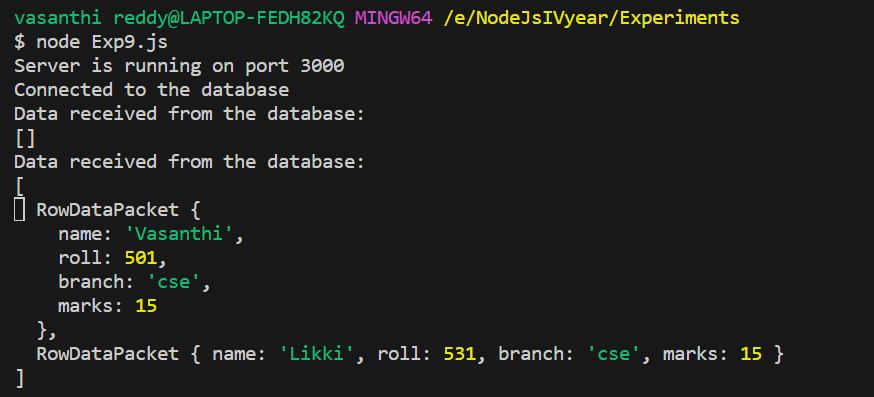
 });

});

// Start the server

app.listen(3000, () => {

 console.log('Server is running on port 3000')});



10.Create a custom error handler to format and send error responses.

const createError = require('http-errors');

const express = require('express');

const app = express();

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

    // Create an error and pass it to the next middleware

    next(createError(500, 'This is a test error.'));

   });

const errorHandler = (err, req, res, next) => {

    if (typeof err === 'string') {

       // If the error is a string, convert it to an Error object

       err = createError(err);

    }

    if (err.status >= 500) {

       // If the error status code is greater than or equal to 500,

       // log the error stack to the console

       console.error(err.stack);

       console.log("Error Generated");

    }

    // Set the response status code to the error status code

    res.status(err.status || 500);

    // Set the response Content-Type to 'application/json'

    res.setHeader('Content-Type', 'application/json');

    // Send the error message in the response body

    res.json({

       message: err.message || 'An error occurred during the request.',

    });

   };

// Use the error handler middleware for all routes

app.use(errorHandler);

// Start the server

app.listen(3000, () => {

 console.log('Server is running on port 3000.');

});

