G Varun

IoT A

21011102033

Web Tech Exc 7

Routing with Express.JS

Frontend (Client) – React

App.js

import React from 'react'

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import HomePage from './components/Home';

import AboutPage from './components/About';

function App() {

  return (

    <Router>

      <Routes>

        <Route path="/" element={<HomePage />} />

        <Route path="/about" element={<AboutPage />} />

      </Routes>

    </Router>

  );

}

export default App;

About.js

import React, { useEffect, useState } from 'react';

const AboutPage = () => {

  const [data, setData] = useState(null);

  useEffect(() => {

    fetch('/api/data')

      .then(response => response.json())

      .then(data => setData(data));

  }, []);

  return (

    <div>

      <h1>About Us</h1>

      <p>This page provides information about us.</p>

      {data && <div>Data from server: {JSON.stringify(data)}</div>}

    </div>

  );

};

export default AboutPage;

Home.js

import { Link } from 'react-router-dom';

import React from 'react'

const HomePage = () => {

  return (

    <div>

      <h1>Welcome to the Home Page</h1>

      <p>This is the home page of our application.</p>

      <Link to="/about">Go to About Page</Link>

    </div>

  );

};

export default HomePage;

Backend – Server (Express)

Server.js

const express = require('express');

const cors = require('cors');

const app = express();

const port = process.env.PORT || 5000;

app.use(cors());

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

  const data = {

    message: 'Hello from the server!',

    // Add more data here as needed

  };

  res.json(data);

});

app.listen(port, () => console.log(`Server is running on port ${port}`));

package.json

"proxy": "http://localhost:5000",

(only this line to be added to the file)

Output:

Client-side

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedServer-side