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

**Class - D15A Batch - C**

**CA Assignment**

1. **Sample program to demonstrate the use of Promise and Callback functions in asynchronous programming in JavaScript.**

function fetchDataFromServer(callback) {

setTimeout(() => {

const data = { message: "Data fetched successfully" };

callback(null, data);

}, 2000);

}

function fetchUsingCallback() {

fetchDataFromServer((error, data) => {

if (error) {

console.error("Error:", error);

} else {

console.log("Callback Data:", data.message);

}

});

}

function fetchUsingPromise() {

return new Promise((resolve, reject) => {

fetchDataFromServer((error, data) => {

if (error) {

reject(error);

} else {

resolve(data);

}

});

});

}

fetchUsingCallback();

fetchUsingPromise()

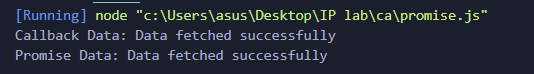
.then((data) => {

console.log("Promise Data:", data.message);

})

.catch((error) => {

console.error("Error:", error); });



1. **Sample Program to demonstrate the use of generator Iterator.**

**Generators**

function\* numberGenerator() {

let i = 0;

while (true) {

yield i++;

}

}

const generator = numberGenerator();

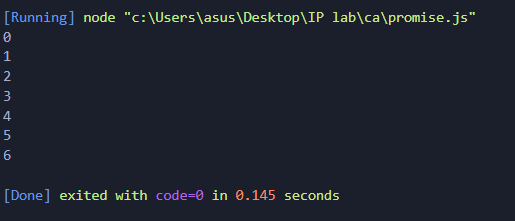
for (let j = 0; j < 5; j++) {

console.log(generator.next().value);

}

console.log(generator.next().value);

console.log(generator.next().value);



**Iterators**

const customIterable = {

data: [1, 2, 3, 4, 5],

[Symbol.iterator]: function () {

let index = 0;

return {

next: () => {

if (index < this.data.length) {

return { value: this.data[index++], done: false };

} else {

return { done: true };

}

},

};

},

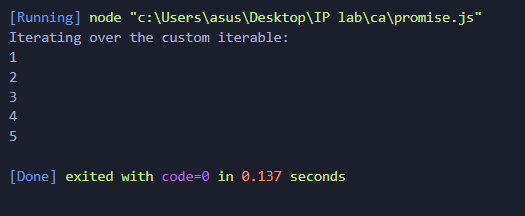
};

console.log("Iterating over the custom iterable:");

for (const item of customIterable) {

console.log(item);

}

****

1. **Sample Program on Routing using REACT,NODE.js, Express.js**

**Using Express:-**

const express = require('express');

const app = express();

const port = 5000;

app.listen(port, () => {

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

});

app.use(express.static('path-to-your-react-app/build'));

// Define a route for the homepage

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

res.sendFile('path-to-your-react-app/build/index.html');

});

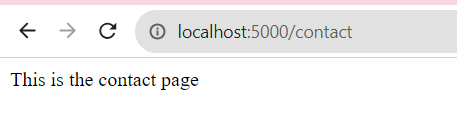
// Define a route for the contact page

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

res.send('This is the contact page');

});





**Using React :-**

**App.js**

import React from 'react';

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

import Home from './Home';

import About from './About';

function App() {

return (

<Router>

<div>

<nav>

<ul>

<li>

<Link to="/">Home</Link>

</li>

<li>

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

</li>

</ul>

</nav>

<hr />

<Route exact path="/" component={Home} />

<Route path="/about" component={About} />

</div>

</Router>

);

}

export default App;

Home.js

import React from 'react';

function Home() {

return <div>Welcome to the homepage!</div>;

}

export default Home;

1. **Explain Where REST APIs are used explain the working of REST API with examples.**

REST (Representational State Transfer) is an architectural style for designing networked applications. REST APIs (Application Programming Interfaces) are a set of constraints and conventions used for building and interacting with web services. REST APIs are widely used in various contexts, including web and mobile applications, cloud services, IoT devices, and more. They allow different software systems to communicate with each other over the internet by following a set of principles and rules.

Here's a high-level explanation of where REST APIs are commonly used and how they work, followed by an example:

### Where REST APIs Are Used:

* Web Services: REST APIs are commonly used to build web services that provide data and functionality to web and mobile applications. For example, a social media platform may expose REST APIs to allow developers to retrieve user profiles or post updates.
* Cloud Services: Many cloud providers offer RESTful APIs for managing cloud resources, such as creating and managing virtual machines, databases, and storage.
* IoT (Internet of Things): REST APIs are used in IoT systems to communicate with devices and sensors, allowing applications to retrieve and control data from IoT devices.
* Third-Party Integrations: REST APIs are used to enable third-party integrations with various services and platforms. For instance, e-commerce platforms often provide APIs for payment gateways or shipping services.
* Authentication and Authorization: REST APIs are used to manage user authentication and authorization. They allow applications to validate user credentials and permissions.
* Data Retrieval: REST APIs are used to fetch data from external sources, such as weather data, stock market data, or news articles.

URL Structure: Design the URL structure to represent the resources and actions. For example:

* GET /todos: Retrieve a list of all todos.
* GET /todos/1: Retrieve the details of a specific todo.
* POST /todos: Create a new todo.
* PUT /todos/1: Update the details of a specific todo.
* DELETE /todos/1: Delete a specific todo.

**5) Explain the use of WebPack in React.**

Webpack is a popular JavaScript module bundler used in React and many other web development projects. It helps manage and optimize the assets and dependencies of your React application. Here's an explanation of how Webpack is used in React:

* **Module Bundling:** Webpack allows you to bundle all your JavaScript, CSS, and other assets into a single or multiple optimized files. This is particularly useful in larger React applications, where you have numerous components and dependencies spread across multiple files. Webpack bundles these files together, which reduces the number of requests your application has to make to the server, resulting in faster load times.
* **Module Resolution:** With Webpack, you can use modern JavaScript features like ES6 modules (import/export statements) and ensure that these modules are resolved and included in the bundle properly. Webpack handles the dependencies and creates a dependency graph.
* **Development Server:** Webpack provides a development server that allows you to run your React application locally during development. This server comes with hot module replacement (HMR) capabilities, which automatically update your application in the browser when you make code changes, saving you from having to manually refresh the page.
* **Loaders:** Webpack uses loaders to process files other than JavaScript. For example, it can use loaders to transpile JSX code using Babel, preprocess CSS with tools like SASS or LESS, optimize images, and more. Loaders help transform and prepare assets before bundling.
* **Environment Configuration:** You can use Webpack to define different build configurations for development, testing, and production environments. This is useful for setting environment-specific variables and optimizations.
* **Asset Optimization:** Webpack can optimize assets like JavaScript and CSS by minimizing and compressing the code, thus reducing the file size and improving load times.