**Program 1**

**Aim: Create a profile card using HTML, CSS, and JavaScript.**

**Source Code:**

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Profile Card</title>

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

</head>

<body>

<div class="profile-card">

<img src="https://th.bing.com/th/id/OIP.7dTfyRneXPY5b7pj0NKuUgHaHa?pid=ImgDet&w=184&h=184&c=7&dpr=1.3" alt="Profile Picture" class="profile-pic">

<h2 class="profile-name">Your Name</h2>

<p class="profile-bio">A short bio about yourself.</p>

<button id="toggle-theme-btn">Toggle Theme</button>

</div>

<script src="script.js"></script>

</body>

</html>

**Style.css:**

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

transition: background-color 0.3s;

}

.profile-card {

text-align: center;

padding: 20px;

border: 1px solid #ccc;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}

.profile-pic {

border-radius: 50%;

margin-bottom: 10px;

width:100px;

height:100px;

align:center;

}

.light-theme {

background-color: #ffffff;

color: #000000;

}

.dark-theme {

background-color: #333333;

color: #ffffff;

}

button {

margin-top: 10px;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

}

**Script.js:**

document.addEventListener("DOMContentLoaded", () => {

const toggleButton = document.getElementById("toggle-theme-btn");

const body = document.body;

// Default theme

body.classList.add("light-theme");

toggleButton.addEventListener("click", () => {

if (body.classList.contains("light-theme")) {

body.classList.replace("light-theme", "dark-theme");

} else {

body.classList.replace("dark-theme", "light-theme");

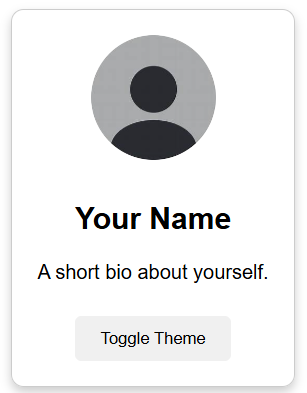
}

});

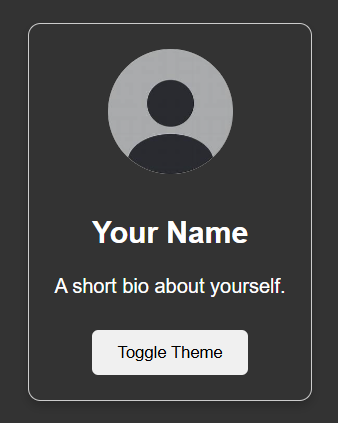
});

**Output:**

After executing the code the profile card is shown below before toggle button is not clicked**:**

****

While Toggle Button is Clicked the background colour is changed the output is shown below:

****

**Program 2**

**Aim : To design /develop a To-do list app by using web development and perform add task and delete operations on the to-do list.**

**Source Code:**

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>To-Do List</title>

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

</head>

<body>

<div class="container">

<h2>✅ To-Do List</h2>

<div class="input-group">

<input type="text" id="taskInput" class="task-input" placeholder="Add Task">

<button onclick="addTask()">Add Task</button>

</div>

<ul id="taskList"></ul>

</div>

<script src="script.js"></script>

</body>

</html>

**Style.css:**

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #e0e0e0;

}

.container {

background: white;

padding: 20px;

border-radius: 15px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

width: 320px;

text-align: center;

}

.input-group {

display: flex;

justify-content: space-between;

margin-bottom: 10px;

}

.task-input {

flex: 1;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

margin-right: 5px;

}

button {

background: #007BFF;

color: white;

border: none;

padding: 10px;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background: #0056b3;

}

ul {

list-style: none;

padding: 0;

}

.task {

display: flex;

align-items: center;

justify-content: space-between;

background: #f9f9f9;

padding: 8px;

margin: 5px 0;

border-radius: 5px;

}

.task input[type="checkbox"] {

margin-right: 10px;

}

.completed span {

text-decoration: line-through;

color: gray;

}

.delete-btn {

background: red;

color: white;

border: none;

padding: 5px;

cursor: pointer;

border-radius: 3px;

}

.delete-btn:hover {

background: darkred;

}

**Script.js:**

function addTask() {

let taskInput = document.getElementById("taskInput");

let taskText = taskInput.value.trim();

if (taskText === "") return;

let taskList = document.getElementById("taskList");

let li = document.createElement("li");

li.classList.add("task");

li.innerHTML = `

<input type="checkbox" onclick="toggleTask(this)">

<span>${taskText}</span>

<button class="delete-btn" onclick="removeTask(this)">Delete</button>

`;

taskList.appendChild(li);

taskInput.value = "";

}

function toggleTask(checkbox) {

checkbox.parentElement.classList.toggle("completed");

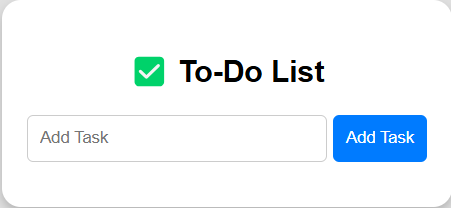
}

function removeTask(button) {

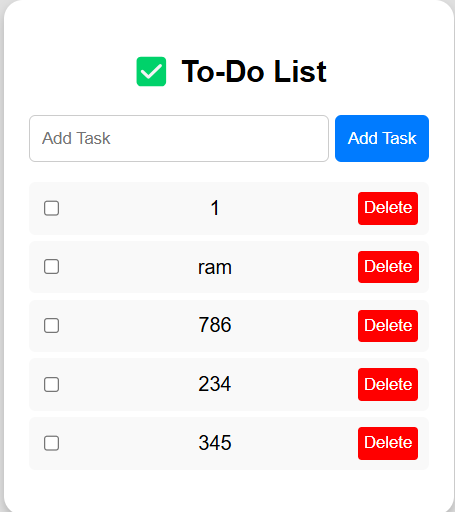
button.parentElement.remove();

}

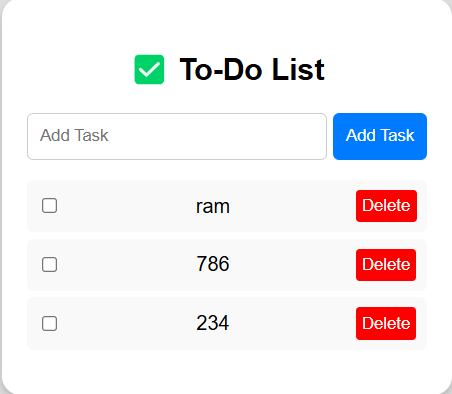
**Ouput:**

****

**Add a tasks to the list are:**

****

**Now, delete the 1,345 lists in the to-do list**

****

**Exercise -II(b)**

**Aim:** To develop / create a To-Do list app using React App.

**Source Code:**

**ToDoList.js:**

import React, { useState } from 'react';

function TodoList() {

  const [tasks, setTasks] = useState([]);

  const [newTask, setNewTask] = useState('');

  const handleInputChange = (event) => {

    setNewTask(event.target.value);

  };

  const handleAddTask = () => {

    if (newTask.trim() !== '') { // Prevent adding empty tasks

      setTasks([...tasks, { text: newTask, completed: false }]);

      setNewTask(''); // Clear input field

    }

  };

  const handleDeleteTask = (index) => {

    const updatedTasks = tasks.filter((\_, i) => i !== index);

    setTasks(updatedTasks);

  };

  const handleCompleteTask = (index) => {

    const updatedTasks = [...tasks];

    updatedTasks[index].completed = !updatedTasks[index].completed;

    setTasks(updatedTasks);

  };

  return (

    <div className="todo-list">

      <div className="header">

        <span className="check-mark">&#10004;</span> {/\* Checkmark symbol \*/}

        <h1>To-Do List</h1>

      </div>

      <div className="add-task">

        <input

          type="text"

          value={newTask}

          onChange={handleInputChange}

          placeholder="Add Task"

        />

        <button onClick={handleAddTask}>Add Task</button>

      </div>

      <ul>

        {tasks.map((task, index) => (

          <li key={index} className={task.completed ? 'completed' : ''}>

            <input

              type="checkbox"

              checked={task.completed}

              onChange={() => handleCompleteTask(index)}

            />

            <span className="task-text">{task.text}</span>

            <button onClick={() => handleDeleteTask(index)}>Delete</button>

          </li>

        ))}

      </ul>

    </div>

  );

}

export default TodoList;

**ToDoList.css:**

.todo-list {

    font-family: sans-serif;

    width: 300px;

    margin: 0 auto;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

  }

  .header {

    display: flex;

    align-items: center;

    margin-bottom: 20px;

  }

  .check-mark {

    font-size: 24px;

    margin-right: 10px;

    color: green; /\* Or your preferred color \*/

  }

  .add-task {

    display: flex;

    margin-bottom: 10px;

  }

  .add-task input {

    flex-grow: 1;

    padding: 8px;

    border: 1px solid #ccc;

    border-radius: 3px;

  }

  .add-task button {

    padding: 8px 12px;

    background-color: #4CAF50; /\* Green \*/

    color: white;

    border: none;

    border-radius: 3px;

    cursor: pointer;

  }

  ul {

    list-style: none;

    padding: 0;

  }

  li {

    display: flex;

    align-items: center;

    padding: 10px;

    border-bottom: 1px solid #eee;

  }

  .completed {

    text-decoration: line-through;

    color: #888;

  }

  .task-text {

    flex-grow: 1;

    margin-left: 10px; /\* Space between checkbox and text \*/

  }

  li button {

    background-color: #f44336; /\* Red \*/

    color: white;

    border: none;

    padding: 6px 10px;

    border-radius: 3px;

    cursor: pointer;

  }

**App.js:**

import React from 'react';

import TodoList from './TodoList'; // Import your TodoList component

function App() {

  return (

    <div className="App">

      <TodoList />  {/\* Use the TodoList component \*/}

    </div>

  );

}

export default App;

**index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App'; // Import your App component

import './styles.css'; // Import the CSS file

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

**Index.html:**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>React App</title>

    <link rel="stylesheet" href="%PUBLIC\_URL%/styles.css">  </head>

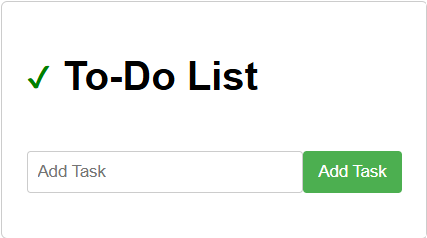
  <body>

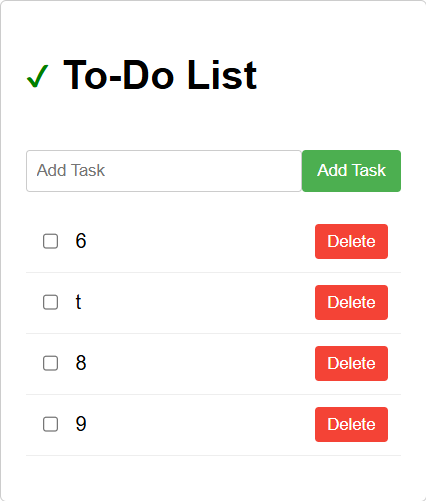
    <div id="root"></div>

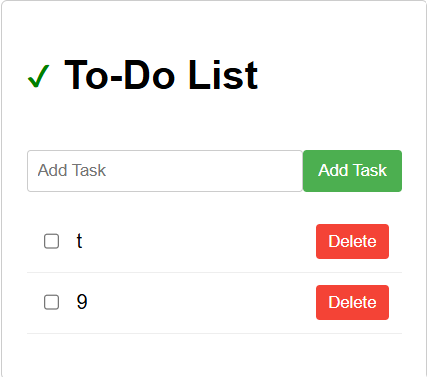
  </body>

</html>

**Output:**

****





**Exercise - III(a)**

**Aim:** To demonstrate the usage of state react component using class.

**Source code:**

**Car.js:**

import React, { Component } from 'react';

class Car extends Component {

    constructor(props) {

        super(props);

        this.state = {

            brand: "Ford",

            model: "Mustang",

            color: "green",

            year: 2009

        };

    }

    changeColor = () => {

        this.setState({ color: "red" });

    }

    render() {

        return (

            <div>

                <h1>My {this.state.brand}</h1>

                <p>

                    It is a {this.state.color}

                    {this.state.model}

                    from {this.state.year}.

                </p>

                <button type="button" onClick={this.changeColor}>

                    Change color

                </button>

            </div>

        );

    }

}

export default Car;

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>My React App</title>

</head>

<body>

<div id="root"></div>

</body>

</html>

**Index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import Car from './Car.js'; // Note the .js extension

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

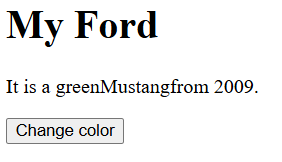
    <React.StrictMode>

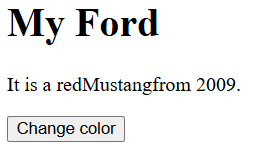
        <Car />

    </React.StrictMode>

);

**Output:**

****

****

**Exercise III(b)**

**Aim:** To demonstrate the usage of state react component using function.

**Source Code:**

**Car.js:**

import React, { useState } from 'react';

function Car() {

  const [car, setCar] = useState({

    brand: "Thar",

    model: "Mustang",

    color: "green",

    year: 2009

  });

  const changeColor = () => {

    setCar({ ...car, color: "Red" }); // Use spread operator to update state correctly

  };

  return (

    <div>

      <h1>My {car.brand}</h1>

      <p>

        It is a {car.color} {car.model} from {car.year}.

      </p>

      <button onClick={changeColor}>Change color</button>

    </div>

  );

}

export default Car;

**index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import Car from './Car'; // Import the functional component

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <Car /> {/\* Render the functional component \*/}

  </React.StrictMode>

);

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>React App</title>

</head>

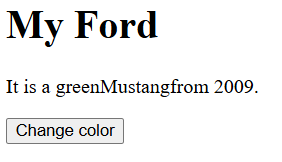
<body>

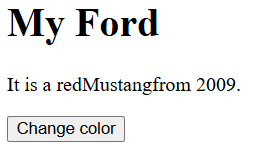
    <div id="root"></div>

    <script src="index.js" type="module"></script> </body>

</html>

**Output:**

****

****

**Exercise IV**

**Aim:** To Create Simple calculator App using react.

**Source Code:**

**Calculator.js:**

import React, { useState } from 'react';

import './Calculator.css'; // Import your CSS file

function Calculator() {

  const [display, setDisplay] = useState('0');

  const handleClick = (value) => {

    if (value === '=') {

      try {

        setDisplay(eval(display).toString()); // Use eval carefully!

      } catch (error) {

        setDisplay('Error');

      }

    } else if (value === 'C') {

      setDisplay('0');

    } else {

      setDisplay((display === '0' && value !== '.') ? value : display + value);

    }

  };

  return (

    <div className="calculator">

      <div className="display">{display}</div>

      <div className="buttons">

        <button onClick={() => handleClick('7')}>7</button>

        <button onClick={() => handleClick('8')}>8</button>

        <button onClick={() => handleClick('9')}>9</button>

        <button className="operator" onClick={() => handleClick('/')}>/</button>

        <button onClick={() => handleClick('4')}>4</button>

        <button onClick={() => handleClick('5')}>5</button>

        <button onClick={() => handleClick('6')}>6</button>

        <button className="operator" onClick={() => handleClick('\*')}>x</button>

        <button onClick={() => handleClick('1')}>1</button>

        <button onClick={() => handleClick('2')}>2</button>

        <button onClick={() => handleClick('3')}>3</button>

        <button className="operator" onClick={() => handleClick('-')}>-</button>

        <button onClick={() => handleClick('0')}>0</button>

        <button onClick={() => handleClick('.')}>.</button>

        <button className="equals" onClick={() => handleClick('=')}>=</button>

        <button className="operator" onClick={() => handleClick('+')}>+</button>

        <button className="clear" onClick={() => handleClick('C')}>C</button>

      </div>

    </div>

  );

}

export default Calculator;

**Calculator.css:**

.calculator {

    width: 250px;

    margin: 50px auto;

    border: 1px solid #ccc;

    border-radius: 5px;

    overflow: hidden; /\* Prevents button overflow \*/

  }

  .display {

    background-color: #eee;

    padding: 10px;

    text-align: right;

    font-size: 20px;

    border-bottom: 1px solid #ccc;

  }

  .buttons {

    display: grid;

    grid-template-columns: repeat(4, 1fr); /\* 4 equal columns \*/

  }

  button {

    padding: 15px;

    font-size: 16px;

    border: 1px solid #ccc;

    background-color: #f0f0f0;

    cursor: pointer;

  }

  button.operator {

    background-color: #f2c14e; /\* Example: Light orange \*/

    color: white;

  }

  button.equals {

    background-color: #4CAF50; /\* Example: Green \*/

    color: white;

  }

  button.clear {

    background-color: #f44336; /\* Example: Red \*/

    color: white;

  }

**App.js:**

import React from 'react';

import ReactDOM from 'react-dom/client'; // Import ReactDOM

import Calculator from './Calculator'; // Import your Calculator component

import './index.css'; // Import your CSS if you have one

function App() {

  return (

    <div className="App">

      <Calculator />

    </div>

  );

}

const root = ReactDOM.createRoot(document.getElementById('root')); // Create a root

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

export default App;

**Index.js:**

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

**Index.html:**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <title>React App</title>

  </head>

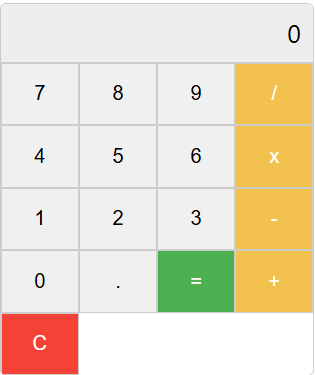
  <body>

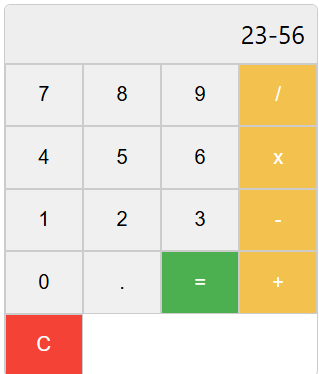
    <div id="root"></div>

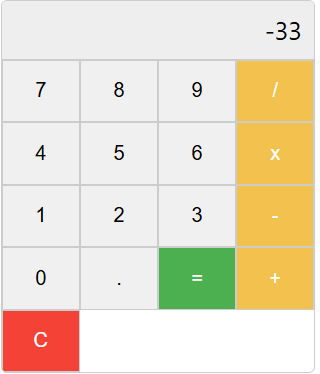
    </body>

</html>

**Output:**

****

****

****

**Exercise-V**

**Aim:** Write a program to create webPage using Routing in react.

**Source Code:**

**App.js:**

import React from 'react';

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

import PageOne from './pages/PageOne';

import PageTwo from './pages/PageTwo';

import PageThree from './pages/PageThree';

function App() {

  return (

    <Router>

      <div style={{ backgroundColor: '#f0f0f0', padding: '20px' }}> {/\* Navbar styling \*/}

        <nav style={{ display: 'flex', gap: '20px' }}> {/\* Button container \*/}

          <button style={{ backgroundColor: 'lightblue', color: 'blue', padding: '10px 15px', border: 'none', borderRadius: '5px', cursor: 'pointer', fontSize: '1em' }}>

            <Link to="/one" style={{ textDecoration: 'none', color: 'blue' }}>

              Go to Page One

            </Link>

          </button>

          <button style={{ backgroundColor: 'lightgreen', color: 'green', padding: '10px 15px', border: 'none', borderRadius: '5px', cursor: 'pointer', fontSize: '1em' }}>

            <Link to="/two" style={{ textDecoration: 'none', color: 'green' }}>

              Go to Page Two

            </Link>

          </button>

          <button style={{ backgroundColor: 'lightsalmon', color: 'white', padding: '10px 15px', border: 'none', borderRadius: '5px', cursor: 'pointer', fontSize: '1em' }}>

            <Link to="/three" style={{ textDecoration: 'none', color: 'white' }}>

              Go to Page Three

            </Link>

          </button>

        </nav>

        <hr />

        <Routes>

          <Route path="/one" element={<PageOne />} />

          <Route path="/two" element={<PageTwo />} />

          <Route path="/three" element={<PageThree />} />

          {/\* You can add a default or "home" route if needed \*/}

          <Route path="/" element={<div><h1>Welcome!</h1><p>Click a button above.</p></div>} />

        </Routes>

      </div>

    </Router>

  );

}

export default App;

**pageOne.js:**

import React from 'react';

function PageOne() {

  return (

    <div>

      <h1>Page One</h1>

      <p>This is the content of Page One.</p>

    </div>

  );

}

export default PageOne;

**PageTwo.js:**

import React from 'react';

function PageTwo() {

  return (

    <div>

      <h1>Page Two</h1>

      <p>This is the content of Page Two.</p>

    </div>

  );

}

export default PageTwo;

**PageThree.js:**

import React from 'react';

function PageThree() {

  return (

    <div>

      <h1>Page Three</h1>

      <p>This is the content of Page Three.</p>

    </div>

  );

}

export default PageThree;

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>React App</title>

</head>

<body>

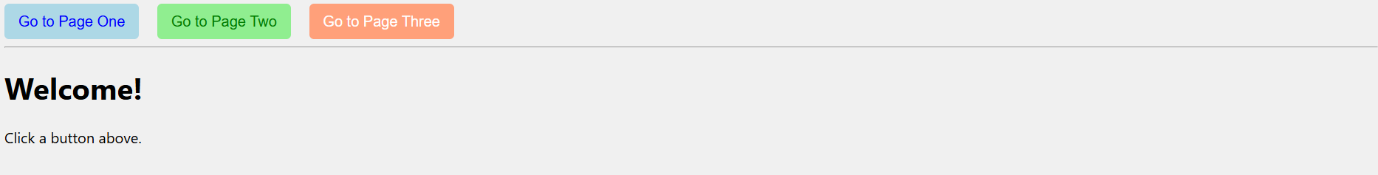
    <div id="root"></div>

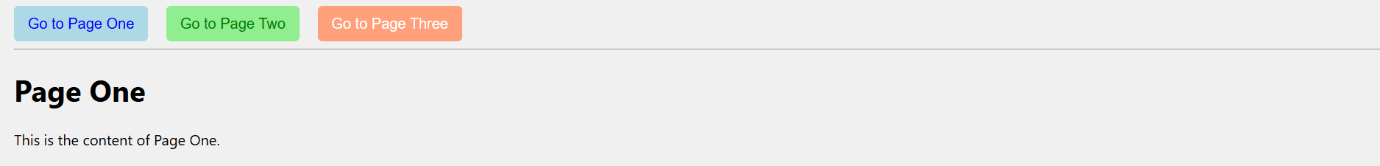
    <script src="/static/js/bundle.js"></script>

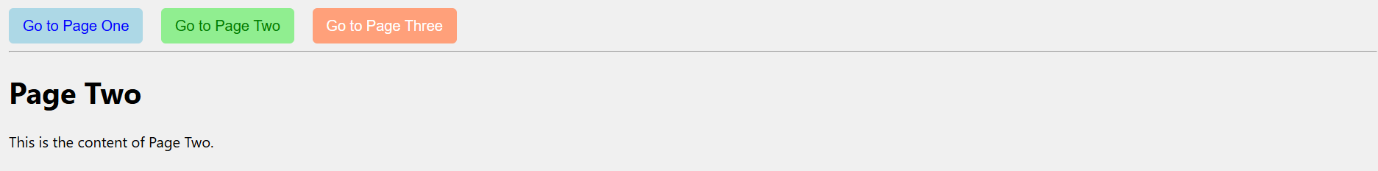
</body>

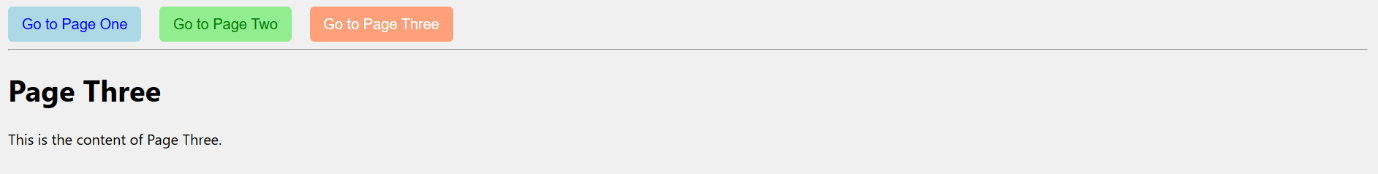
</html>

**Output:**

****



****

****

**Exercise-VI**

**Aim:** Write a program to create Registration Form using React.

**Source Code:**

**Form.js:**

import { useState } from "react";

export default function Form(){

  const [username,setname]=useState("");

  const [email,setemail]=useState("");

  const [pass,setpass]=useState("");

  const [sub,setsub]=useState(false);

  const [err,seterr]=useState(false);

  const handlename=(e)=>{

    setname(e.target.value);

  };

  const handleemail=(e)=>{

    setemail(e.target.value);

  };

  const handlepass=(e)=>{

    setpass(e.target.value);

  };

  const handlesubmit=(e)=>{

    e.preventDefault()

    if (!username||!email||!pass){

      seterr(true)

      setsub(false)

    }

    else{

      setsub(true);

      seterr(false);

    }

  };

  const successmsg=()=>{

    if(sub){

      return(<h3>User <b>{username}</b> Registered Successfully</h3>);

    }

    else if(err){

      return(<h3>Please Fill All The Fields</h3>);

    }

};

return(

  <div className="main">

  <div><h1> Registration Form</h1></div>

  <div className="form">

  <form>

      <label>userName :<input type="text" onChange={handlename} value={username} ></input></label>

      <label>Email :<input type="email" onChange={handleemail} value={email}></input></label>

      <label>Password :<input type="password" onChange={handlepass} value={pass}></input></label>

    <button onClick={handlesubmit}>Submit</button>

  </form>

  </div>

  <footer>

  <div className="msg">

      {successmsg()}

    </div>

    </footer>

  </div>

);}

**App.js:**

import React from 'react';

import Form from './Form'; // Assuming your Form component is in Form.js

function App() {

  return (

    <div>

      <Form />

    </div>

  );

}

export default App;

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>User Registration Form</title>

    <link rel="stylesheet" href="index.css"> </head>

<body>

    <div id="root">

        </div>

    </body>

</html>

**Index.css:**

body {

  font-family: sans-serif;

  display: flex;

  justify-content: center;

  align-items: center;

  min-height: 100vh;

  background-color:blanchedalmond;

  margin: 0;

}

.main {

  background-color:rgb(53, 0, 128);

  padding: 30px;

  border-radius: 8px;

  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

  width: 400px;

}

.main h1 {

  text-align: center;

  color:gold;

  margin-bottom: 20px;

}

.form label {

  display: block;

  margin-bottom: 10px;

  color: green;

  font-weight: bold;

}

.form input[type="text"],

.form input[type="email"],

.form input[type="password"] {

  width: calc(100% - 12px);

  padding: 8px;

  border: 1px solid #ccc;

  border-radius: 4px;

  margin-top: 5px;

  box-sizing: border-box;

}

.form button {

  background-color:red;

  color:brown;

  padding: 10px 15px;

  border: none;

  border-radius: 4px;

  cursor: pointer;

  font-size: 16px;

  margin-top: 20px;

  width: 100%;

}

.form button:hover {

  background-color: #0056b3;

}

footer {

  margin-top: 20px;

  text-align: center;

}

.msg h3 {

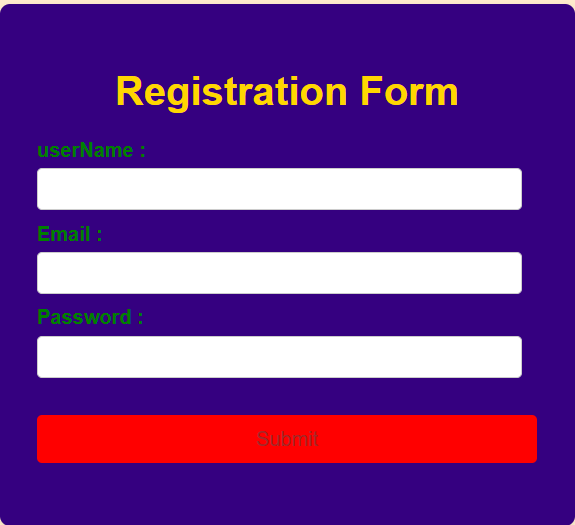
  color:orange; /\* Default success color \*/

}

.msg h3:nth-child(2) { /\* Targeting the error message specifically \*/

  color: red;

}

******Output:**



**Exercise-VII**

**Aim:** To design a Login Form and Logout webpage using Express.js and Node.js in React.

**Source Code:**

**Login.ejs:**

const express = require('express');

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

const path = rcequire('path');

const app = express();

const port = 3000;

// Middleware to parse URL-encoded bodies (for form data)

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

app.use(bodyParser.json());

// Set EJS as the view engine

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

// Dummy user data (replace with a database in a real application)

const users = [];

// Routes

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

res.redirect('/register'); // Redirect to the registration page initially

});

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

res.render('register');

});

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

const { username, email, password } = req.body;

// Basic validation (you should add more robust validation)

if (!username || !email || !password) {

return res.send('All fields are required.');

}

// Check if the username already exists (in a real app, query your database)

if (users.find(user => user.username === username)) {

return res.send('Username already exists.');

}

// Store the new user (in a real app, save to your database)

users.push({ username, email, password });

console.log('Registered user:', { username, email });

res.redirect('/login');

});

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

res.render('login');

});

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

const { username, password } = req.body;

// Find the user (in a real app, query your database)

const user = users.find(u => u.username === username && u.password === password);

if (user) {

// In a real application, you would set up a session here

res.render('dashboard', { username: user.username });

} else {

res.send('Invalid username or password.');

}

});

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

res.render('signup'); // This page can have similar logic to register if needed

});

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

// In a real application, you would check if the user is logged in

if (users.length > 0) { // Basic check for demonstration

res.render('dashboard', { username: 'User' }); // Replace 'User' with the actual logged-in username

} else {

res.redirect('/login');

}

});

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

**Register.ejs:**

<!DOCTYPE html>

<html>

<head>

<title>Register</title>

</head>

<body>

<h1>Register</h1>

<form action="/register" method="POST">

<div>

<label for="username">Username:</label>

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

</div>

<br>

<div>

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

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

</div>

<br>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

<br>

<button type="submit">Register</button>

</form>

<p>Already have an account? <a href="/login">Login</a></p>

<p>Don't have an account? <a href="/signup">Sign Up</a></p>

</body>

</html>

**Login.ejs:**

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<form action="/login" method="POST">

<div>

<label for="username">Username:</label>

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

</div>

<br>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

<br>

<button type="submit">Login</button>

</form>

<p>Don't have an account? <a href="/register">Register</a></p>

<p>Need to sign up? <a href="/signup">Sign Up</a></p>

</body>

</html>

**Signup.ejs:**

<!DOCTYPE html>

<html>

<head>

<title>Sign Up</title>

</head>

<body>

<h1>Sign Up</h1>

<form action="/register" method="POST"> <div>

<label for="username">Username:</label>

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

</div>

<br>

<div>

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

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

</div>

<br>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

<br>

<button type="submit">Sign Up</button>

</form>

<p>Already have an account? <a href="/login">Login</a></p>

<p>Want to register? <a href="/register">Register</a></p>

</body>

</html>

**Dashboard.ejs:**

<!DOCTYPE html>

<html>

<head>

<title>Dashboard</title>

</head>

<body>

<h1>Welcome to the Dashboard, <%= username %>!</h1>

<p>This is your personalized dashboard.</p>

<a href="/">Go to Home</a>

<br>

<a href="/register">Register New User</a>

<br>

<a href="/login">Login</a>

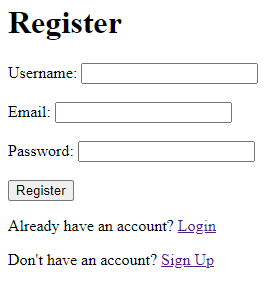
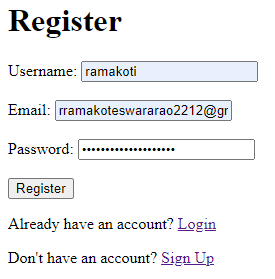
<br>

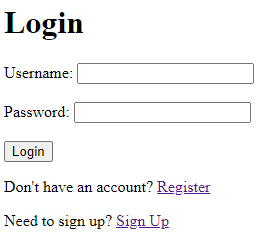
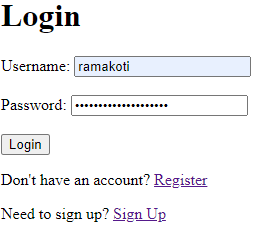
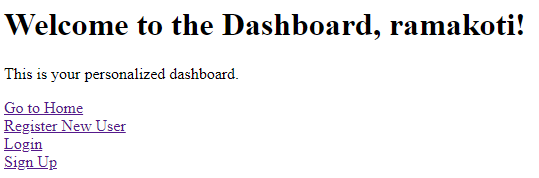
<a href="/signup">Sign Up</a>

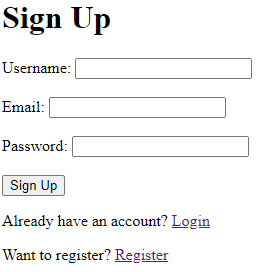
</body>

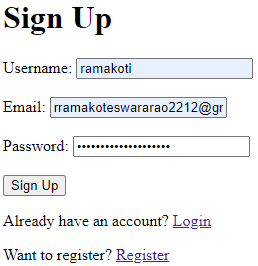
</html>

**Output:**






**Exercise-VIII**

**Aim:** Write a program to demonstrate handling data I/O(Buffer,Stream modules) .

**Source code:**

**buffer\_read\_write.js:**

//Buffer Reading

buf = new

Buffer.allocUnsafe(26); for (var

i = 0 ; i < 26 ; i++)

{

buf[i] = i + 97;

}

console.log("Buffer Reading example...")

console.log( buf.toString('ascii')); // outputs: abcdefghijklmnopqrstuvwxyz

console.log( buf.toString('ascii',0,5));

//Buffer Writing

console.log("")

console.log("Buffer Writing example...")

buf = new

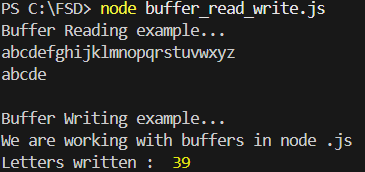
Buffer.allocUnsafe(256);

len = buf.write("We are working with buffers in node .js");

console.log(buf.toString())

console.log("Letters written : ",len);

**Output:**

****

**stream\_read.js:**

//ReadStream

//Reading stream from a txt file

var fs = require("fs");

var data = '';

// Create a readable stream

var readerStream = fs.createReadStream('output.txt');

// Set the encoding to be utf8.

readerStream.setEncoding('UTF8');

// Handle stream events --> data, end, and error

readerStream.on('data', function(chunk)

{

data += chunk;

});

readerStream.on('end',function()

{

console.log(data);

});

readerStream.on('error', function(err)

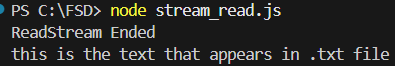
{

console.log(err.stack);

});

console.log("ReadStream Ended");

**Output:**

****

**stream\_write.js:**

//WriteStream

//writing into stream using fs module

var fs = require("fs");

var data = 'this is the text that appears in .txt file';

// Create a writable stream

var writerStream = fs.createWriteStream('output.txt');

// Write the data to stream with encoding to be utf8

 writerStream.write(data,'UTF8');

// Mark the end of file

writerStream.end();

// Handle stream events --> finish, and error

writerStream.on('finish', function()

{

console.log("Write completed...Open output.txt file to verfy");

});

writerStream.on('error', function(err)

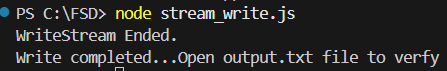
{

console.log(err.stack);

});

console.log("WriteStream Ended.");

**Output:**

****

**Exercise-IX**

**Aim:** Write a program to demonstrate accessing file system from Node.js application(fs module) **.**

**Source code:**

**file\_read\_write\_sync.js**

//file writing in synchronous

 var fs = require('fs');

var veglist = ['carrots','brinjal', 'beetroots', 'raddish','chilli',

'capsicum']; fd = fs.openSync('vegetables.txt', 'w');

console.log("Writing a file in sync-type...")

while (veglist.length)

{

veg = veglist.pop() + " ";

var bytes = fs.writeSync(fd, veg, null, null);

console.log("Wrote ---> %s %d bytes", veg, bytes);

}

fs.closeSync(fd);

//file reading in synchronous

 var fs = require('fs');

fd = fs.openSync('vegetables.txt', 'r');

var vegR = "";

console.log("")

console.log("Reading a file in sync-type...")

do

{

var buf = new Buffer.allocUnsafe(5);

buf.fill();

var byt = fs.readSync(fd, buf, null,

5); console.log("read %d bytes",

byt); vegR += buf.toString();

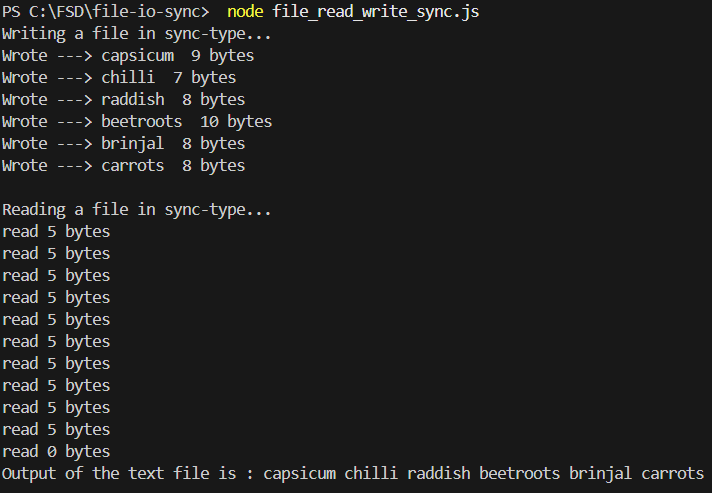
}while (byt > 0);

fs.closeSync(fd);

console.log("Output of the text file is :",vegR);

console.log("");

**Output:**

****