IT3511 - FULL STACK WEB DEVELOPMENT LABORATORY

LIST OF EXPERIMENTS: TOTAL: 60 PERIODS

1. Develop a portfolio website for yourself which gives details about yourself for a potential recruiter.
2. Create a web application to manage the TO-DO list of users, where users can login and manage their to-do items.
3. Create a simple micro blogging application (like twitter) that allows people to post their content which can be viewed by people who follow them.
4. Create a food delivery website where users can order food from a particular restaurant listed in the website.
5. Develop a classifieds web application to buy and sell used products.
6. Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days using react
7. Develop a simple dashboard for project management where the statuses of various tasks are available. New tasks can be added and the status of existing tasks can be changed among Pending, In Progress or Completed.
8. Develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions.

Ex No : 1 **DATE :**	**Develop a portfolio website for yourself which gives details about yourself for a potential recruiter.**

AIM:

To develop a portfolio website using ReactJS.

ALGORITHM:

Step 1: Setup
- Create a new React app using create-react-app.

Step 2: Component Creation
- Inside the src folder, create a components directory.
- For each section of our portfolio (Header, About, Projects, Contact), create a separate .js file inside the components directory.

Step 3: Component Content
- Define the content for each component using JSX.
- Add details about skills, experiences, projects, and contact information.

Step 4: Styling
- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components
- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing
- Run the development server (npm start) to see our portfolio website in the browser.

PROGRAM:

Header.js

```
import React from 'react';
import './Header.css';

const Header = () => {
  return (
    <header className="header">
      <h1>RANI</h1>
      <p>Frontend Developer</p>
    </header>
  );
};
```

```
export default Header;
```

About.js

```
import React from 'react';
import './About.css';

const About = () => {
  return (
    <section className="about">
      <h2>About Me</h2>
      <p>
        I am a passionate frontend developer with a strong interest in creating
        engaging and user-friendly web applications. I enjoy turning complex
        problems into simple, beautiful, and intuitive solutions.
      </p>
    </section>
  );
};

export default About;
```

Projects.js

```
import React from 'react';

const Projects = () => {
  return (
    <section className="projects">
      <h2>Event Management Website</h2>
    <h2>I am Building  a platform for managing events. Users can create events, send invitations, and manage RSVPs.
Include event details, location, and scheduling.</h2>
      {/* Add your project details here */}
    </section>
  );
};

export default Projects;
```

Contact.js:

```
import React from 'react';

const Contact = () => {
  return (
    <section className="contact">
      <h2>Contact Me</h2>
      <p>Email: Rani@jkkn.ac.in</p>
      <p>LinkedIn: Rani.com@LinkedIn</p>
      {/* Add other contact details */}
  </section>
 );
};

export default Contact;
```

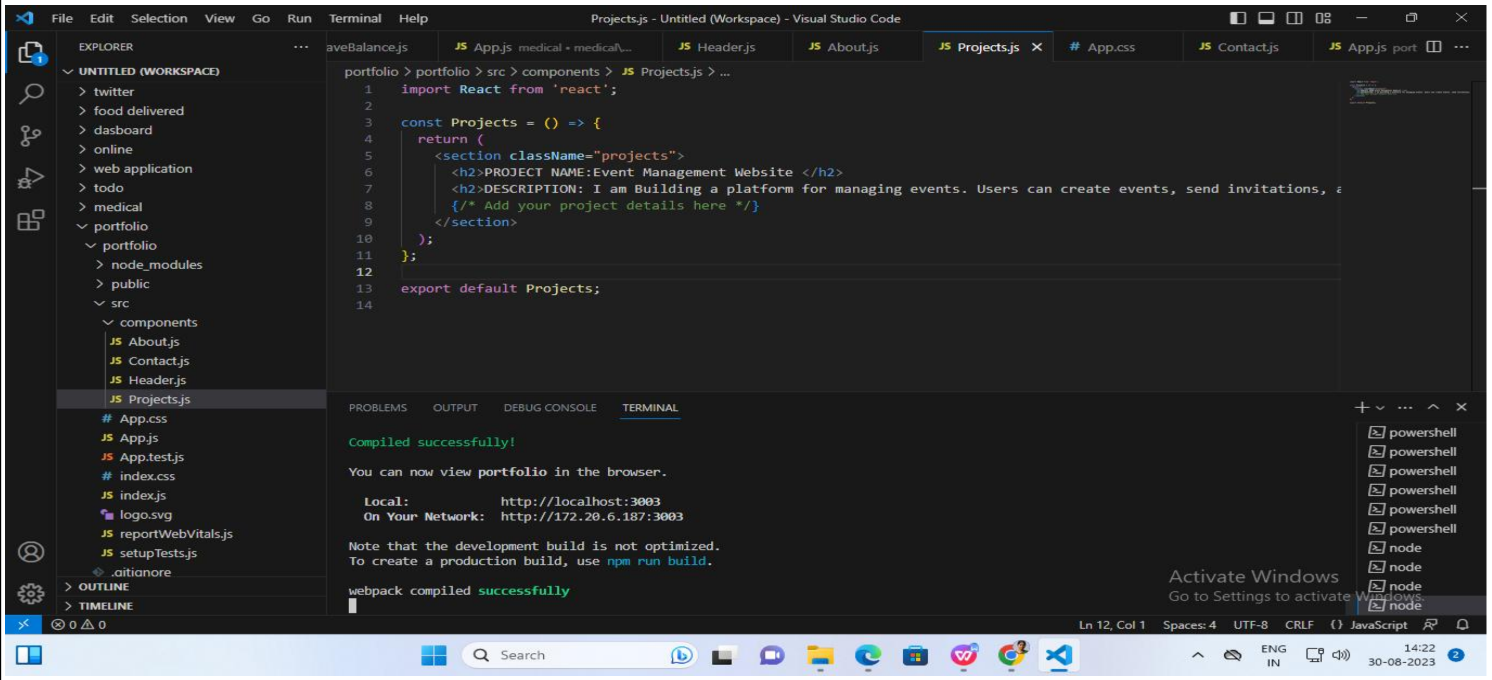
App.js:

```
import React from 'react';
import './App.css';
import Header from './components/Header';
import About from './components/About';
import Projects from './components/Projects';
import Contact from './components/Contact';

function App() {
 return (
  <div className="app">
   <Header />
   <About />
   <Projects />
   <Contact />
  </div>
 );
}

export default App;
```

App.css:

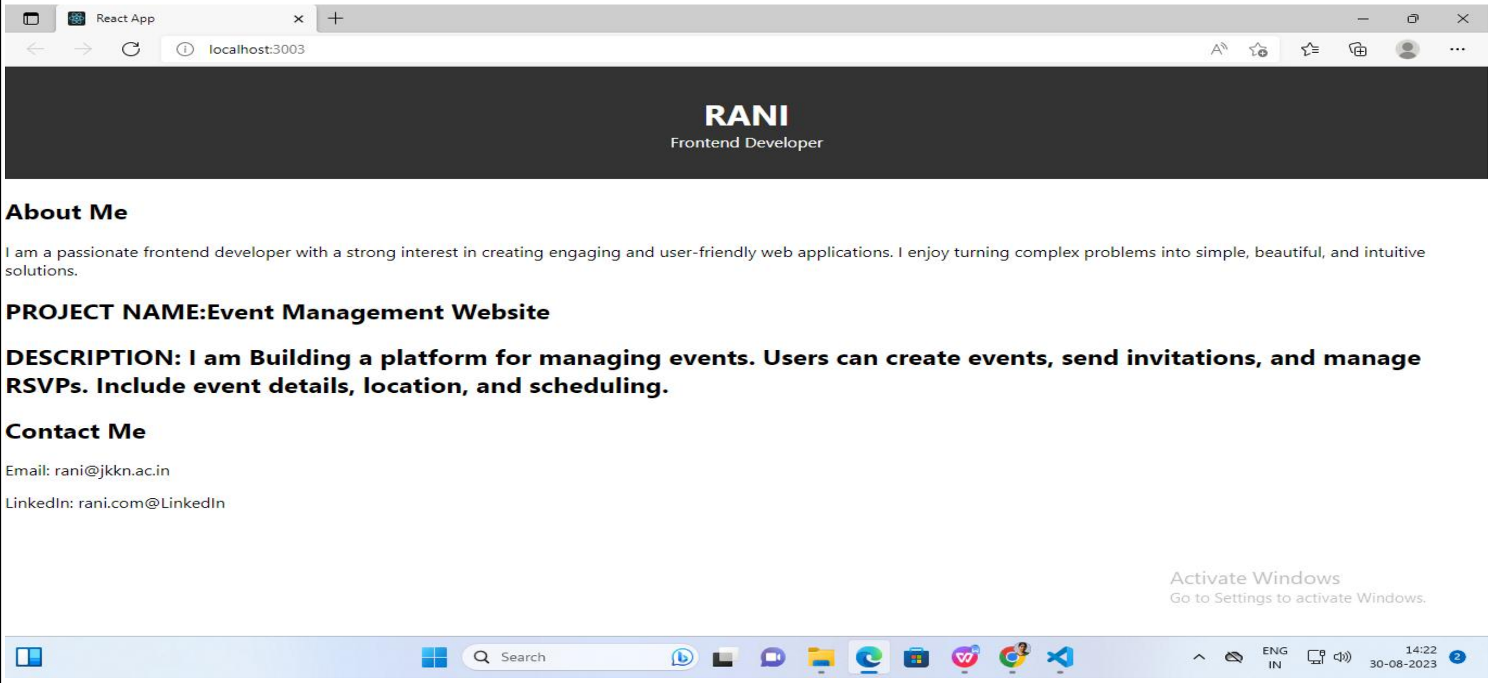
```
.header {
 background-color: #333;
 color: #fff;
 text-align: center;
 padding: 2rem;
}

.header h1 {
 margin: 0;
}

.header p {
 margin: 0;
}
```

OUTPUT:



```
import React from 'react';

const Projects = () => {
  return (
    <section className="projects">
      <h2>PROJECT NAME:Event Management Website </h2>
      <h2>DESCRIPTION: I am Building a platform for managing events. Users can create events, send invitations, a
      {/* Add your project details here */}
    </section>
  );
};

export default Projects;
```

```
Compiled successfully!

You can now view portfolio in the browser.

  Local:            http://localhost:3003
  On Your Network:  http://172.20.6.187:3003

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



RANI

Frontend Developer

About Me

I am a passionate frontend developer with a strong interest in creating engaging and user-friendly web applications. I enjoy turning complex problems into simple, beautiful, and intuitive solutions.

PROJECT NAME:Event Management Website

DESCRIPTION: I am Building a platform for managing events. Users can create events, send invitations, and manage RSVPs. Include event details, location, and scheduling.

Contact Me

Email: rani@jkkn.ac.in

LinkedIn: rani.com@LinkedIn

RESULT:

Thus, the development of a portfolio website using ReactJS was successfully done and verified.

Ex No : 2 DATE :	**Create a web application to manage the TO-DO list of users, where users can login and manage their to-do items**

AIM:

To Create a web application to manage the TO-DO list of users, where users can login and manage their to-do items using React.

ALGORITHM:

Step 1: Set Up

- Create a new React app with necessary components.
- Set up states to track login status, to-do list, and new to-do input.

Step 2:Login Component (Login.js)

- Display a form with username and password fields.
- Manage user input using state.

Step 3:TodoItem Component (TodoItem.js)

- Display an individual to-do item.

Step 4:TodoList Component (TodoList.js)

- Show a list of to-do items.
- Manage the list using state.
- Add new items to the list.

Step 5:App Component (App.js)

- Track the logged-in state using state.
- Show either login or to-do list based on the state.
- Pass the login callback to the Login component.

Step 6:Testing and Running:

- Start the app to see the login screen.

PROGRAM:

Login.js:

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

const Login = ({ onLogin }) => {
const [username, setUsername] = useState('');
const [password, setPassword] = useState('');
const handleLogin = () => {
// Simulated login logic
if (username === 'user' && password === 'password') {
onLogin(true);
} else {
alert('Invalid credentials');
}
};
return (
<div>
<h2>Login</h2>
<input
type="text"
placeholder="Username"
value={username}
onChange={(e) => setUsername(e.target.value)}
/>
<input
type="password"
placeholder="Password"
value={password}
onChange={(e) => setPassword(e.target.value)}
/>
<button onClick={handleLogin}>Login</button>
</div>
);
};

export default Login;
```

TodoItem.js:

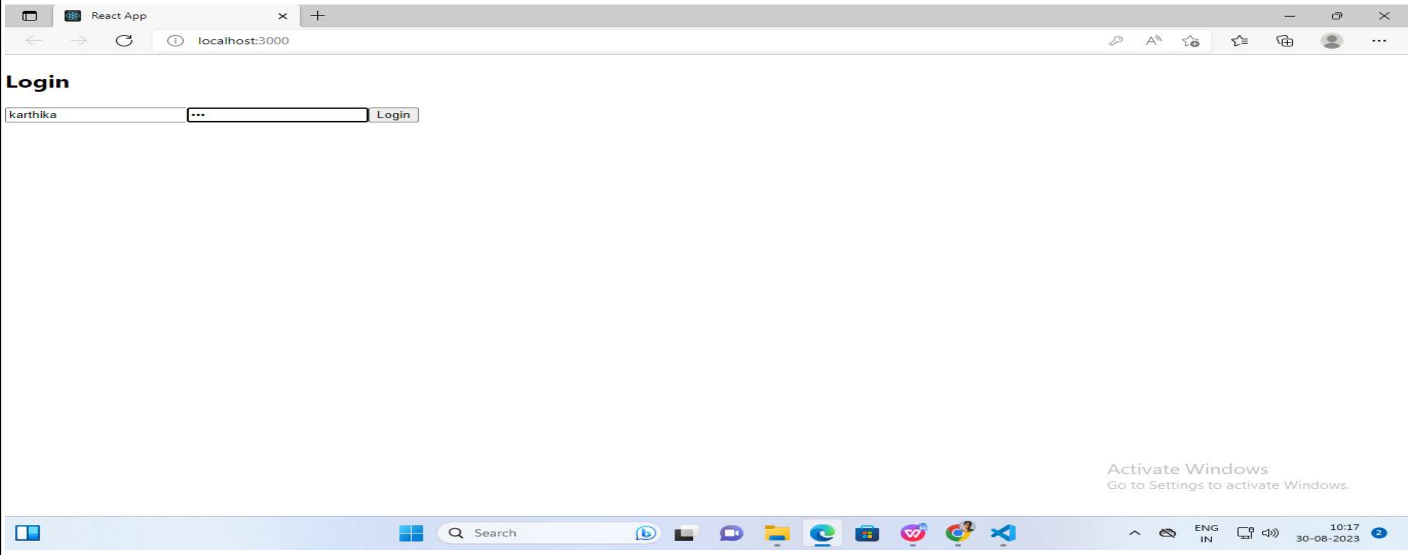
```
import React from 'react';
const TodoItem = ({ text }) => {
  return <li>{text}</li>;
};
export default TodoItem;
```

TodoList.js:

```
import React, { useState } from 'react';
import TodoItem from './TodoItem';

const TodoList = () => {
  const [todos, setTodos] = useState([]);
  const [newTodo, setNewTodo] = useState('');

  const addTodo = () => {
    if (newTodo.trim() !== '') {
      setTodos([...todos, newTodo]);
      setNewTodo('');
    }
```

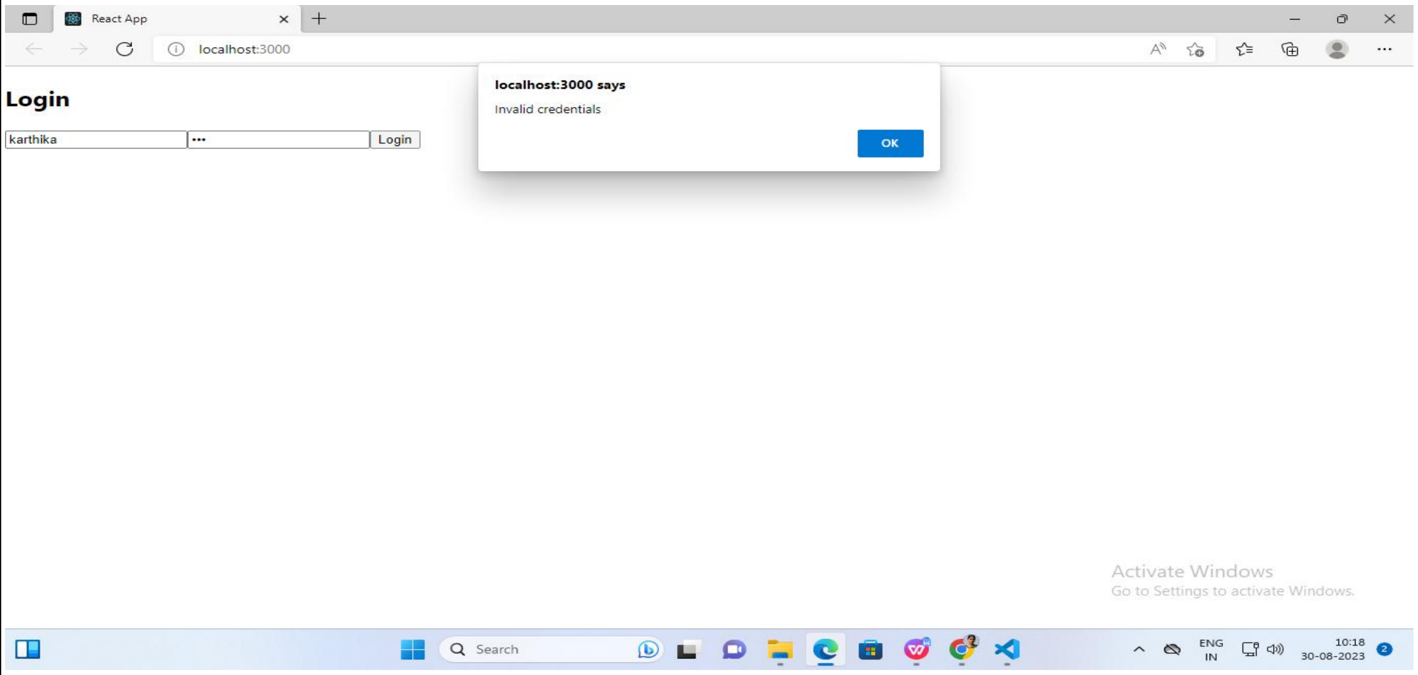
```
 };

 return (
   <div>
     <h2>To-Do List</h2>
     <ul>
       {todos.map((todo, index) => (
         <TodoItem key={index} text={todo} />
       ))}
     </ul>
     <input
       type="text"
       placeholder="New Todo"
       value={newTodo}
       onChange={(e) => setNewTodo(e.target.value)}
     />
     <button onClick={addTodo}>Add</button>
   </div>
 );
};

export default TodoList;
```

Output:



Login

karthika ••• Login



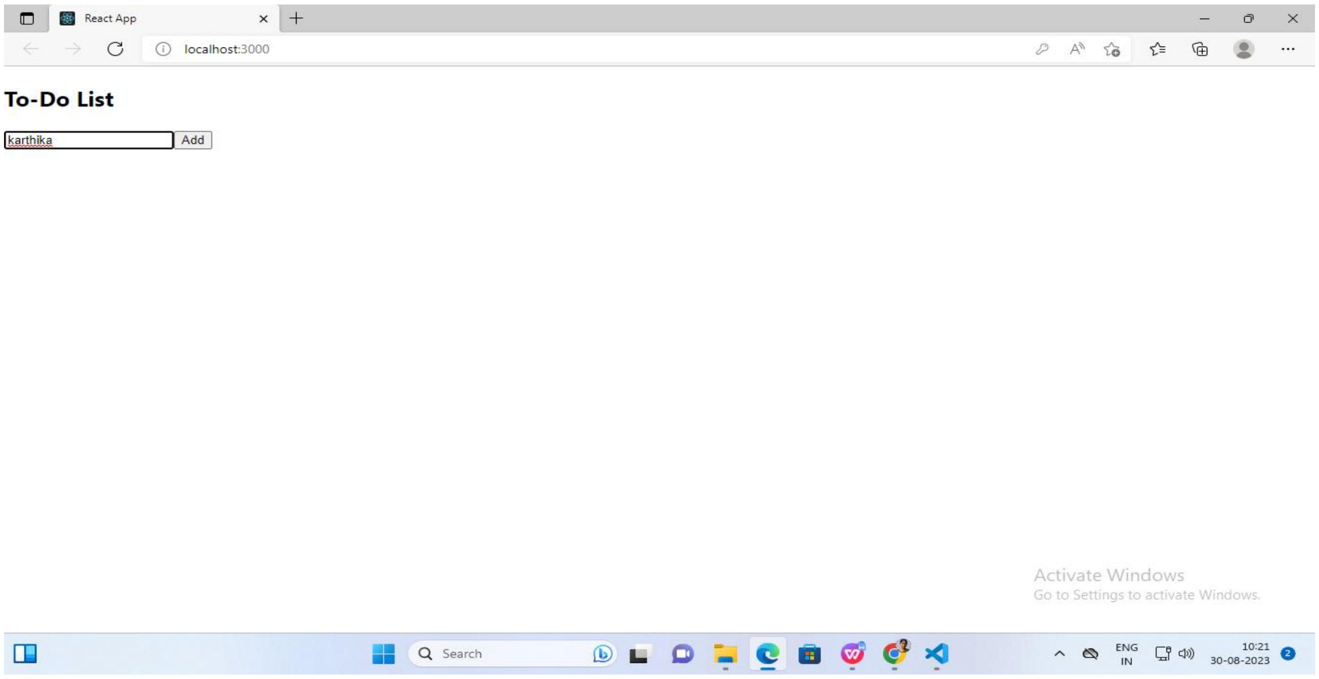
Login

karthika ••• Login

localhost:3000 says

Invalid credentials

OK



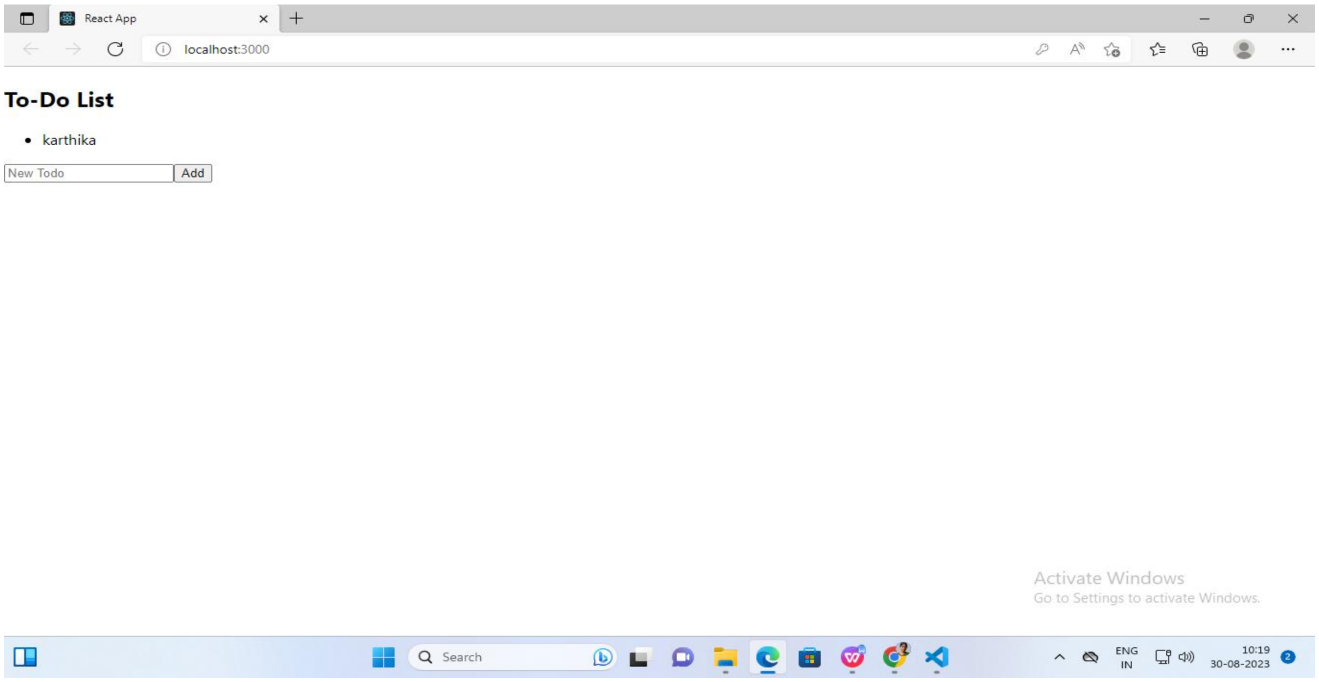
React App

localhost:3000

To-Do List

karthika Add

Activate Windows
Go to Settings to activate Windows.



React App

localhost:3000

To-Do List

- karthika

New Todo Add

Activate Windows
Go to Settings to activate Windows.

RESULT:

Thus, the execution of create a web application to manage the TO-DO list of users, where users can login and manage their to-do items using React was successfully create and executed.

Ex No : 3 DATE :	**Create a simple micro blogging application (like twitter) that allows people to post their content which can be viewed by people who follow them.**

AIM:

To Create a Simple Micro Blogging Application Using React.

ALGORITHM:

Step 1: Setup

- Create a new React app using create-react-app.

Step 2: Component Creation

- Inside the src folder, create a components directory.
- For each section of our micro blogging (Header, Usercard, Postcard, Dashboard), create a separate .js file inside the components directory.

Step 3: Component Content

- Define the content for each component using JSX.
- Add details about card information.

Step 4: Styling

- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components

- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing

- Run the development server (npm start) to see micro blogging application website in the browser.

PROGRAM:

Header.js

```
import React from 'react';

  const Header = () => {
    return (
      <header>
        <h1>Microblogging App</h1>
      </header>
    );
  };

  export default Header
```

UserCard.js

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

const UserCard = ({ user, onFollowToggle }) => {
  const [isFollowing, setIsFollowing] = useState(false);

  const handleFollowToggle = () => {
    setIsFollowing(!isFollowing);
    onFollowToggle(user.id, !isFollowing);
  };

  return (
    <div className="user-card">
      <h3>{user.name}</h3>
      <p>{user.bio}</p>
      <button onClick={handleFollowToggle}>
        {isFollowing ? 'Unfollow' : 'Follow'}
      </button>
    </div>
  );
};

export default UserCard;
```

PostCard.js

```
import React from 'react';

const PostCard = ({ post }) => {
  return (
    <div className="post-card">
      <h3>{post.title}</h3>
      <p>{post.content}</p>
      <p>By {post.author}</p>
    </div>
  );
};

export default PostCard;
```

Dashboard.js

```
import React, { useState } from 'react';
import UserCard from './UserCard';
import PostCard from './PostCard';

const Dashboard = () => {
  // Static data for users and posts
  const [users] = useState([
    { id: 1, name: 'John Doe', bio: 'Frontend Developer' },
    { id: 2, name: 'Jane Smith', bio: 'Backend Developer' },
  ]);

  const [posts] = useState([
    { id: 1, title: 'My First Post', content: 'Hello, world!', author: 'John Doe' },
    { id: 2, title: 'Exciting News', content: 'Just launched my new website!', author: 'Jane Smith' },
  ]);

  const [following, setFollowing] = useState([]);

  const handleFollowToggle = (userId, isFollowing) => {
    if (isFollowing) {
      setFollowing([...following, userId]);
    } else {
      setFollowing(following.filter((id) => id !== userId));
    }
  };

  return (
    <div className="dashboard">
      <h2>Dashboard</h2>
      <div className="user-list">
        {users.map((user) => (
          <UserCard
            key={user.id}
            user={user}
            onFollowToggle={handleFollowToggle}
          />
        ))}
      </div>
      <h3>Posts</h3>
      <div className="post-list">
        {posts
          .filter((post) => following.includes(users.find((user) => user.name === post.author).id))
          .map((post) => (
            <PostCard key={post.id} post={post} />
          ))}
      </div>
    </div>
  );
};

export default Dashboard;
```

App.js

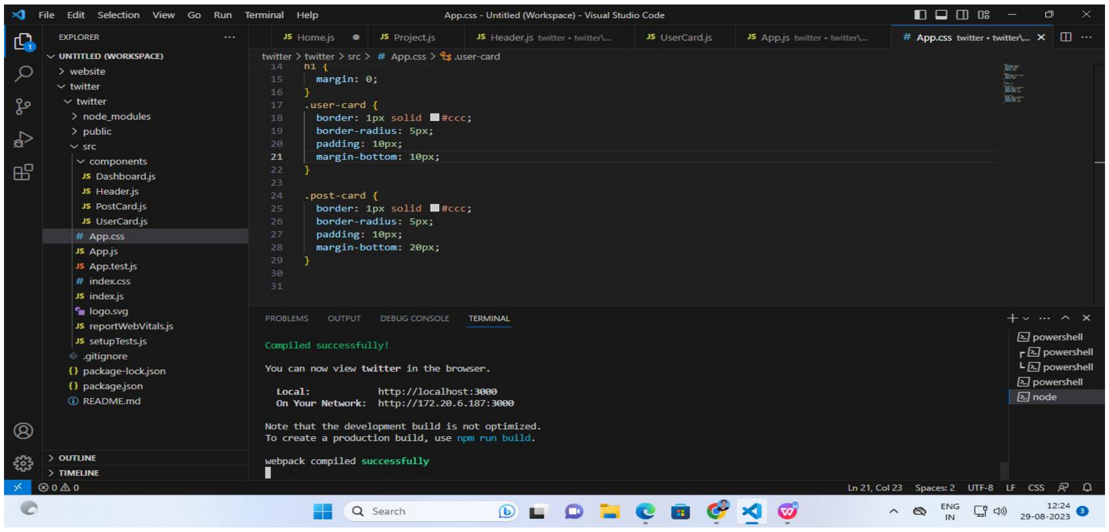
```
import React from 'react';
import './App.css';
import Header from './components/Header';
import Dashboard from './components/Dashboard';

const App = () => {
  return (
    <div className="app">
      <Header />
      <Dashboard />
    </div>
  );
};

export default App;
```

App.css

```
.app {
  text-align: center;
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
}
```

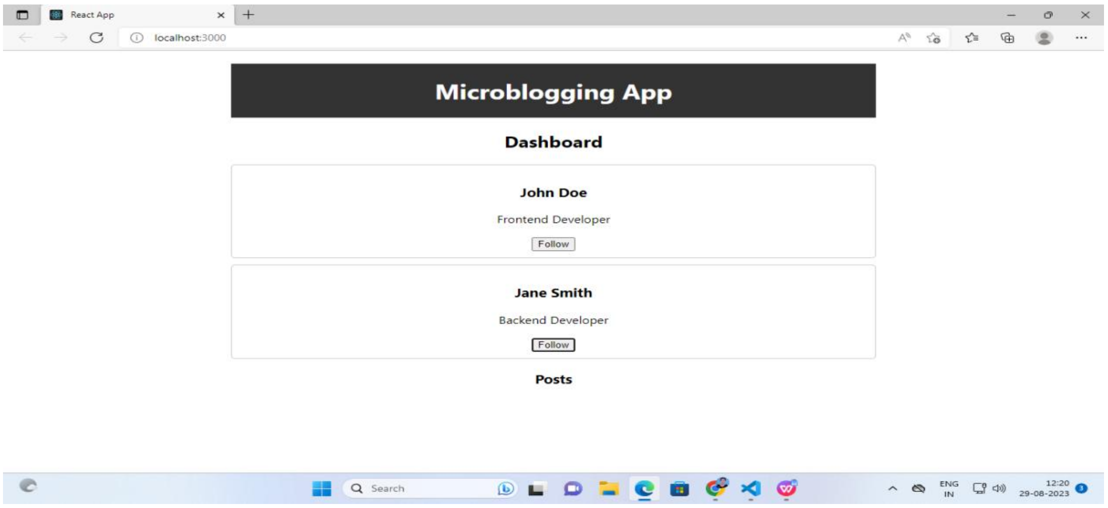
```
header {
  background-color: #333;
  color: #fff;
  padding: 20px;
}

h1 {
  margin: 0;
}
.user-card {
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  margin-bottom: 10px;
}

.post-card {
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  margin-bottom: 20px;
}
```

OUTPUT:



```
h1 {
  margin: 0;
}
.user-card {
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  margin-bottom: 10px;
}

.post-card {
  border: 1px solid #ccc;
  border-radius: 5px;
  padding: 10px;
  margin-bottom: 20px;
}
```

```
Compiled successfully!

You can now view twitter in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.20.6.187:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Microblogging App

Dashboard

John Doe

Frontend Developer

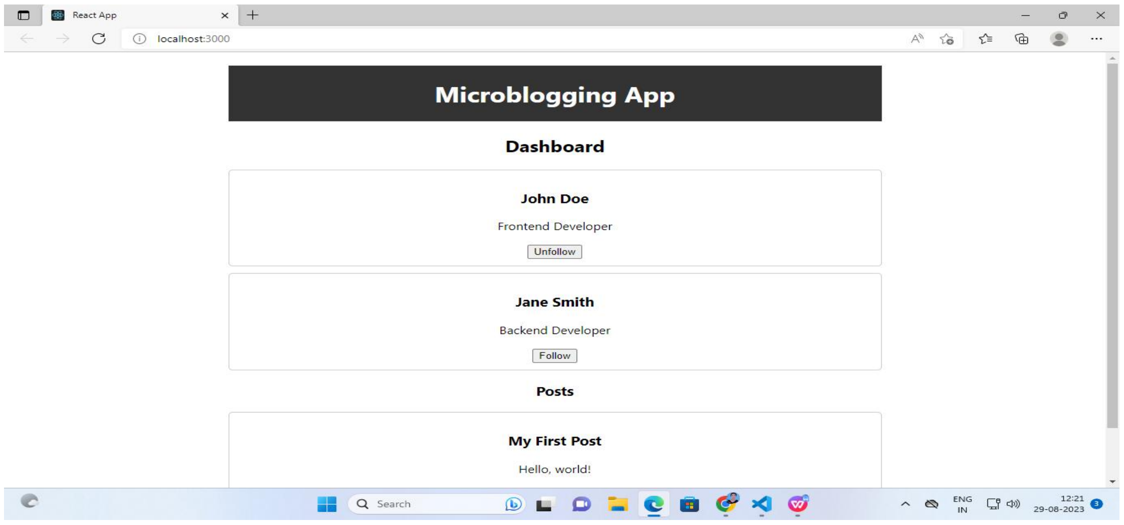
Follow

Jane Smith

Backend Developer

Follow

Posts



Microblogging App

Dashboard

John Doe

Frontend Developer

Unfollow

Jane Smith

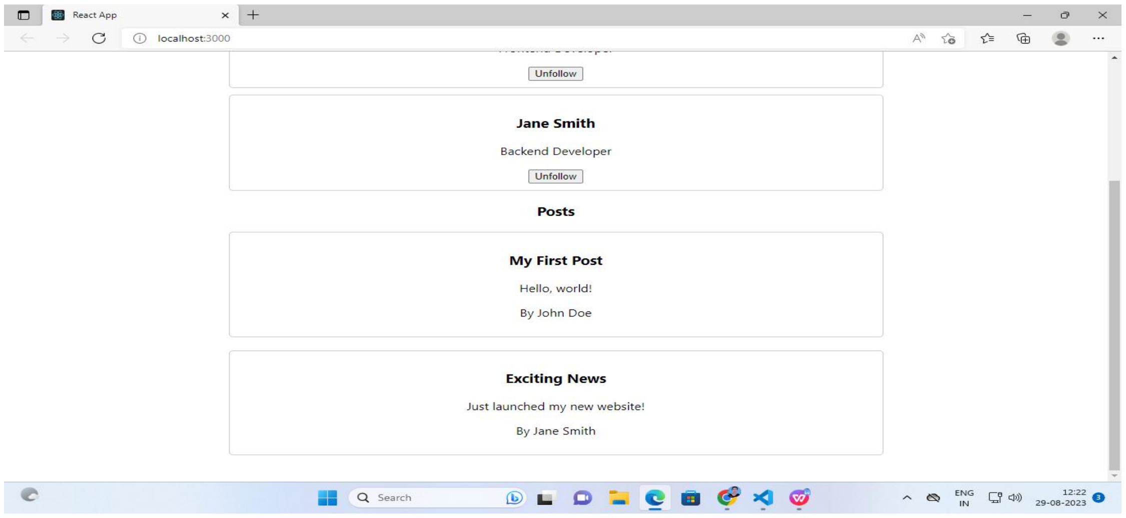
Backend Developer

Follow

Posts

My First Post

Hello, world!



Unfollow

Jane Smith

Backend Developer

Unfollow

Posts

My First Post

Hello, world!

By John Doe

Exciting News

Just launched my new website!

By Jane Smith

RESULT:

Thus, the execution of create a simple micro blogging application using React was successfully create and executed.

Ex No : 4 DATE :	**Create a food delivery website where users can order food from a particular restaurant listed in the website.**

AIM:

To Create a food delivery website where users can order food from a particular restaurant listed in the website using ReactJS.

ALGORITHM:

Step 1: Setup
- Create a new React app using create-react-app.

Step 2: Component Creation
- Inside the src folder, create a components directory.
- For each section of our food delivery website (ResturantList, menu, card), create a separate .js file inside the components directory.

Step 3: Component Content
- Define the content for each component using JSX.
- Add details about card information.

Step 4: Styling
- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components
- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing
- Run the development server (npm start) to see our food delivery website in the browser.

PROGRAM:

RestaurantList.js:

```
import React, { useState } from 'react';
import Menu from './Menu';

const RestaurantList = () => {
  const restaurants = [
    { id: 1, name: 'Restaurant A', menu: ['Burger', 'Pizza', 'Pasta'] },
    { id: 2, name: 'Restaurant B', menu: ['Sushi', 'Tempura', 'Ramen'] },
  ];

  const [selectedRestaurant, setSelectedRestaurant] = useState(null);

  const handleRestaurantSelect = (restaurant) => {
    setSelectedRestaurant(restaurant);
  };

  return (
    <div>
      <h2>Choose a Restaurant</h2>
      <ul>
        {restaurants.map((restaurant) => (
          <li key={restaurant.id}>
            <button onClick={() => handleRestaurantSelect(restaurant)}>
              {restaurant.name}
            </button>
          </li>
        ))}
      </ul>
      {selectedRestaurant && <Menu menu={selectedRestaurant.menu} />}
    </div>
  );
};

export default RestaurantList;
```

Menu.js:

```
import React, { useState } from 'react';
import Cart from './Cart';

const Menu = ({ menu }) => {
  const [cart, setCart] = useState([]);

  const addToCart = (item) => {
    setCart([...cart, item]);
  };

  return (
    <div>
      <h3>Menu</h3>
      <ul>
        {menu.map((item, index) => (
          <li key={index}>
            {item}
            <button onClick={() => addToCart(item)}>Add to Cart</button>
          </li>
        ))}
      </ul>
      <Cart cart={cart} />
    </div>
  );
};
export default Menu;
```

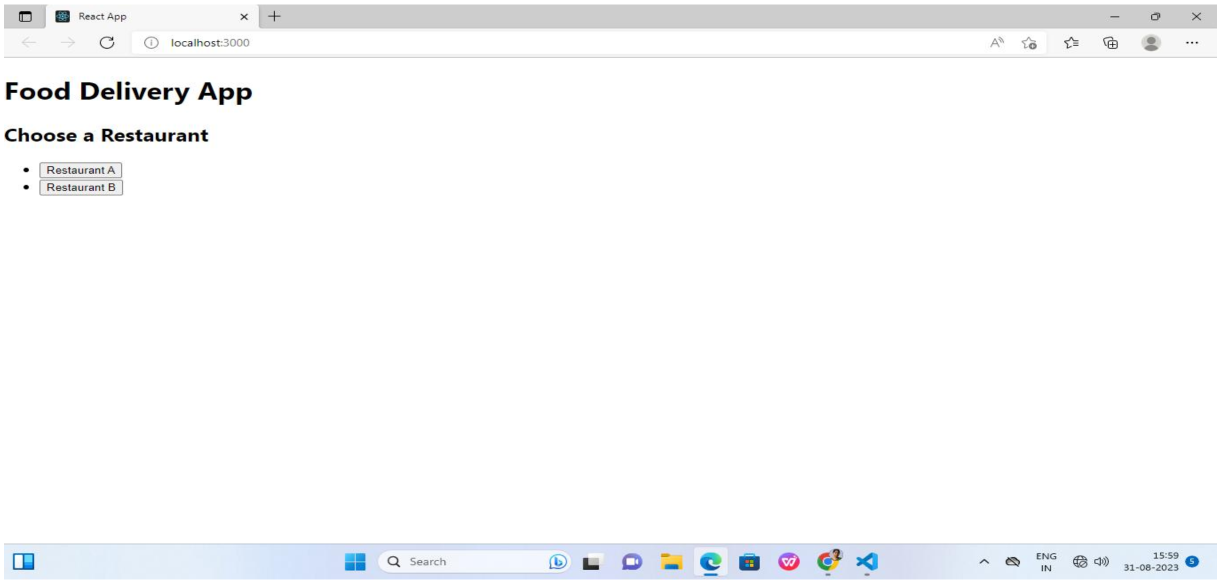
Cart.js:

```
import React from 'react';

const Cart = ({ cart }) => {
  const calculateTotal = () => {
    const itemPrices = {
      Burger: 10,
      Pizza: 12,
      Pasta: 8,
      Sushi: 15,
      Tempura: 8,
      Ramen: 10,
    };

    return cart.reduce((total, item) => total + itemPrices[item], 0);
  };

  return (
    <div>
      <h3>Cart</h3>
      <ul>
        {cart.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
      <p>Total: ${calculateTotal()}</p>
    </div>
  );
};

export default Cart;
```

App.js:

```
import React from 'react';
import './App.css';
import RestaurantList from './components/RestaurantList';

function App() {
  return (
    <div className="app">
      <h1>Food Delivery App</h1>
      <RestaurantList />
    </div>
  );
}

export default App;
```

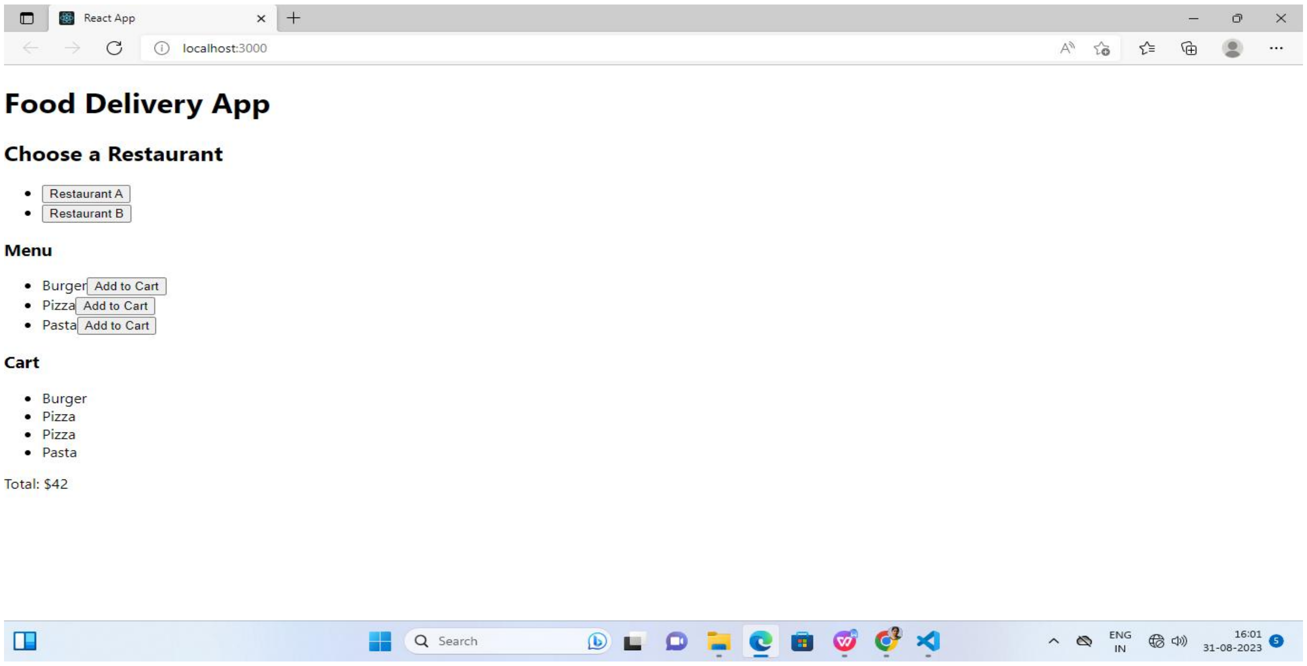
OUTPUT:



Food Delivery App

Choose a Restaurant

- Restaurant A
- Restaurant B



Food Delivery App

Choose a Restaurant

- Restaurant A
- Restaurant B

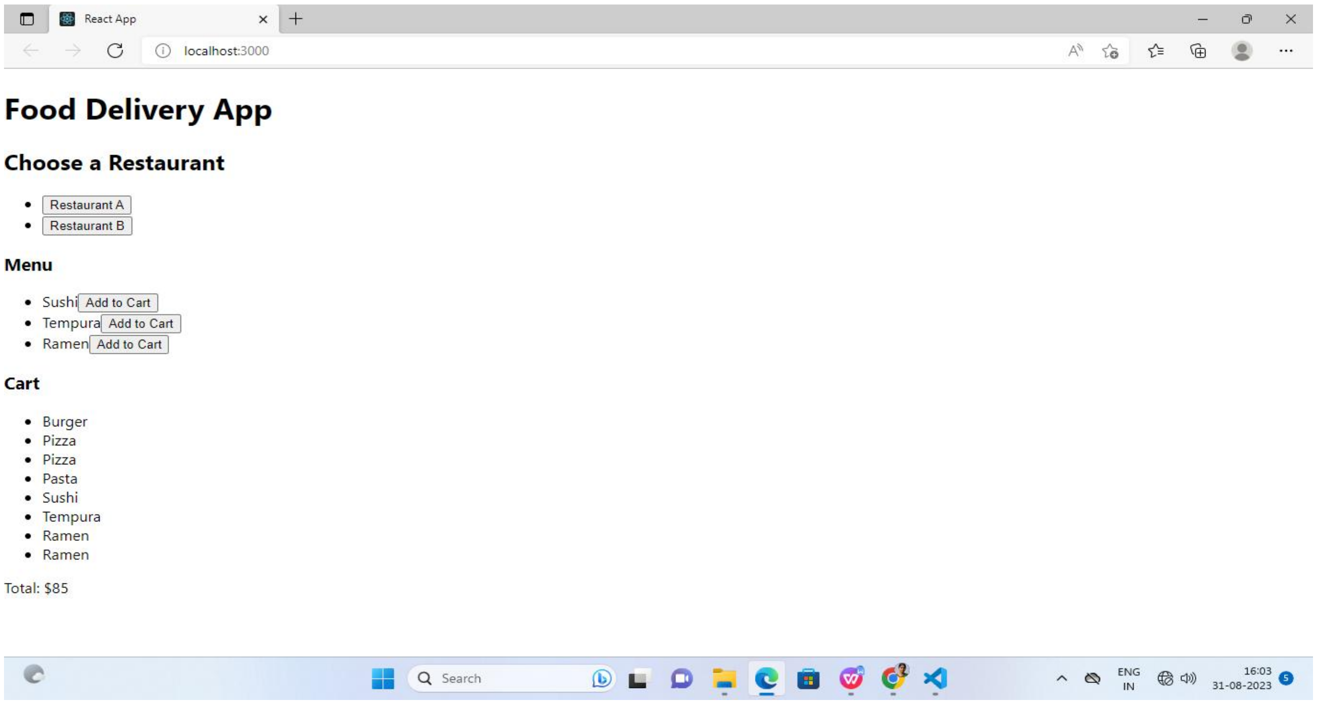
Menu

- Burger Add to Cart
- Pizza Add to Cart
- Pasta Add to Cart

Cart

- Burger
- Pizza
- Pizza
- Pasta

Total: $42



React App

localhost:3000

Food Delivery App

Choose a Restaurant

- Restaurant A
- Restaurant B

Menu

- Sushi Add to Cart
- Tempura Add to Cart
- Ramen Add to Cart

Cart

- Burger
- Pizza
- Pizza
- Pasta
- Sushi
- Tempura
- Ramen
- Ramen

Total: $85

RESULT:

Thus, the execution of create a food delivery website using React was successfully done and executed.

Ex No : 5 DATE :	**Develop a classifieds web application to buy and sell used products.**

AIM:

To develop a classifieds web application to buy and sell used products using ReactJS.

ALGORITHM:

Step 1: Setup

- Create a new React app using create-react-app.

Step 2: Component Creation

- Inside the src folder, create a components directory.
- For each section of our buy and sell website (Header.js,ProductList.js), create a separate .js file inside the components directory.

Step 3: Component Content

- Define the content for each component using JSX.
- Add details about card information.

Step 4: Styling

- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components

- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing

- Run the development server (npm start) to see buy and sell website in the browser.

PROGRAM:

Header.js

```
import React from 'react';

const Header = () => {
  return (
    <header>
      <h1>Classifieds - Buy & Sell Used Products</h1>
    </header>
  );
};

export default Header;
```

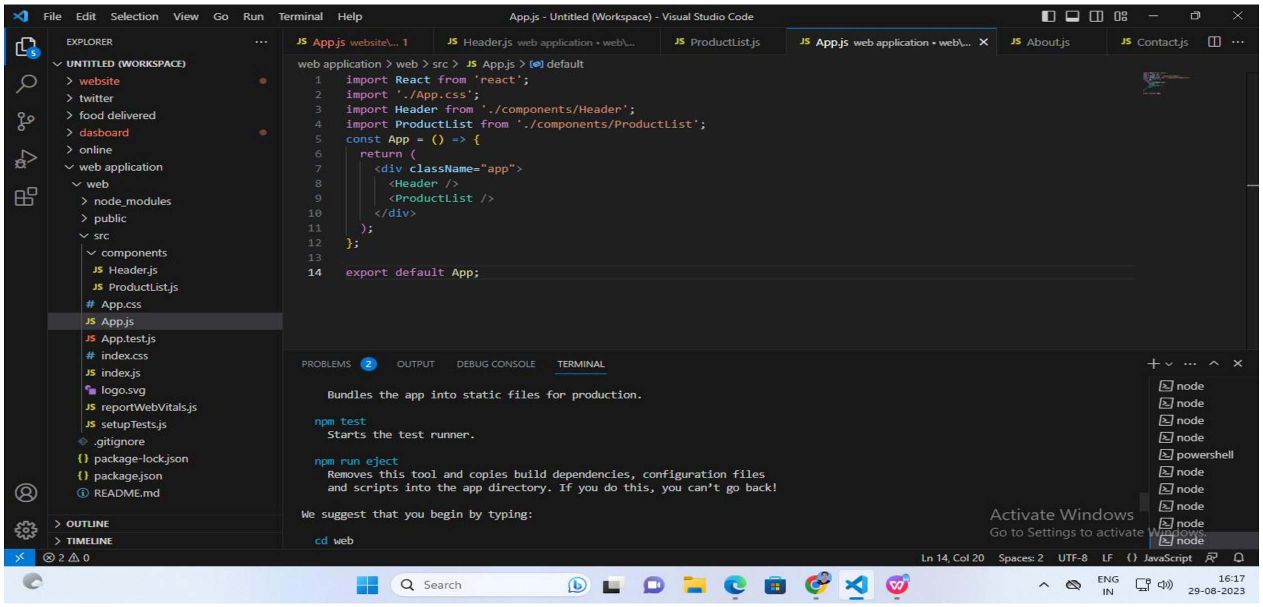
ProductList.js

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

const ProductList = () => {
  const [products, setProducts] = useState([
    { id: 1, title: 'Used Laptop', description: 'Good condition, 1-year-old laptop for sale.', price: 300 },
    { id: 2, title: 'Old Smartphone', description: 'Selling a used smartphone in working condition.', price: 100 },
    // Add more products here...
  ]);

  return (
    <div className="product-list">
      <h2>Used Products Listing</h2>
      <ul>
        {products.map((product) => (
          <li key={product.id}>
            <h3>{product.title}</h3>
            <p>{product.description}</p>
            <p>Price: ${product.price}</p>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ProductList;
```

App.js

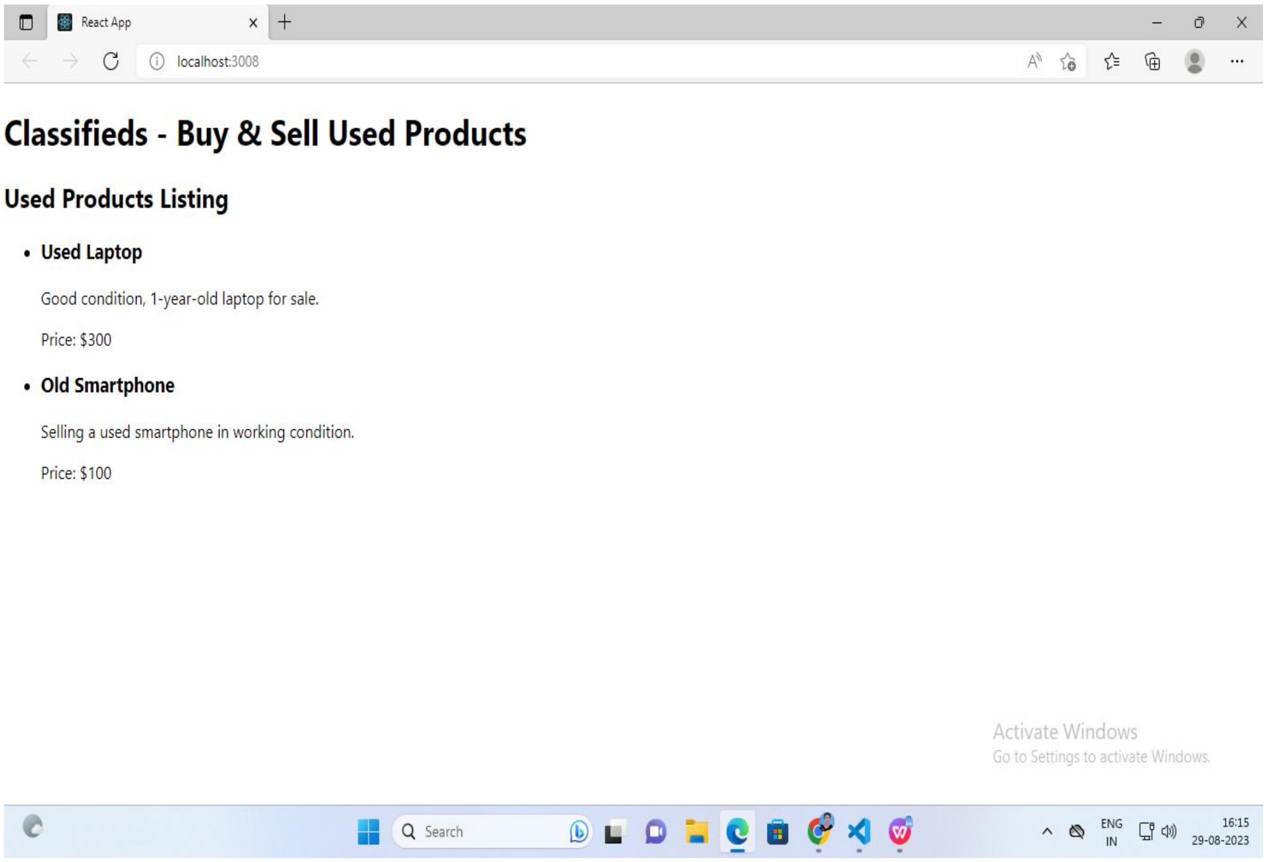
```
import React from 'react';
import './App.css';
import Header from './components/Header';
import ProductList from './components/ProductList';
const App = () => {
  return (
    <div className="app">
      <Header />
      <ProductList />
    </div>
  );
};

export default App;
```

OUTPUT:



```
import React from 'react';
import './App.css';
import Header from './components/Header';
import ProductList from './components/ProductList';
const App = () => {
  return (
    <div className="app">
      <Header />
      <ProductList />
    </div>
  );
};

export default App;
```

```
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd web
```



React App

localhost:3008

Classifieds - Buy & Sell Used Products

Used Products Listing

- **Used Laptop**

  Good condition, 1-year-old laptop for sale.

  Price: $300

- **Old Smartphone**

  Selling a used smartphone in working condition.

  Price: $100

RESULT:

Thus, the execution of create a buy and sell used products website using react was successfully done and executed.

EXP No : 6 DATE :	**Develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave. They also can view the available number of days**

AIM:

To develop a leave management system for an organization where users can apply different types of leaves such as casual leave and medical leave, they also can view the available number of days using React.

ALGORITHM:

Step 1: Setup

- Create a new React application with necessary components.
- Initialize leave balances for different types of leaves in the state.

Step 2:LeaveApplication Component (LeaveApplication.js)

- Render a form for applying for different types of leaves.
- Manage user input using state variables for selected leave type and days.
- Implement a submit button with a logic to handle leave application.

Step 3:LeaveBalance Component (LeaveBalance.js)

- Render the available leave balance for different types of leaves.
- Iterate through leave balance data and display each leave type along with its balance.

Step 4:App Component (App.js):

- Manage the overall application state.
- Create a function to handle leave application logic.
- Pass relevant data to LeaveApplication and LeaveBalance components.

Step 5:Apply Leave Logic

- When the user submits a leave application, validate the input.
- Check if the selected leave type exists and if the days are valid.
- If the conditions are met, update the leave balance accordingly.

PROGRAM:

LeaveApplication.js

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

const LeaveApplication = ({ leaveTypes, onApply }) => {
  const [selectedLeaveType, setSelectedLeaveType] = useState('');
  const [days, setDays] = useState('');

  const handleSubmit = () => {
    if (selectedLeaveType && days) {
      onApply(selectedLeaveType, days);
      setSelectedLeaveType('');
      setDays('');
    }
  };

  return (
    <div>
      <h2>Leave Application</h2>
      <div>
        <label>
          Leave Type:
          <select value={selectedLeaveType} onChange={(e) => setSelectedLeaveType(e.target.value)}>
            <option value="">Select leave type</option>
            {leaveTypes.map((type) => (
              <option key={type} value={type}>
                {type}
              </option>
            ))}
          </select>
        </label>
      </div>
      <div>
        <label>
          Number of Days:
          <input type="number" value={days} onChange={(e) => setDays(e.target.value)} />
        </label>
      </div>
      <button onClick={handleSubmit}>Apply</button>
    </div>
  );
};

export default LeaveApplication;
```

LeaveBalance.js

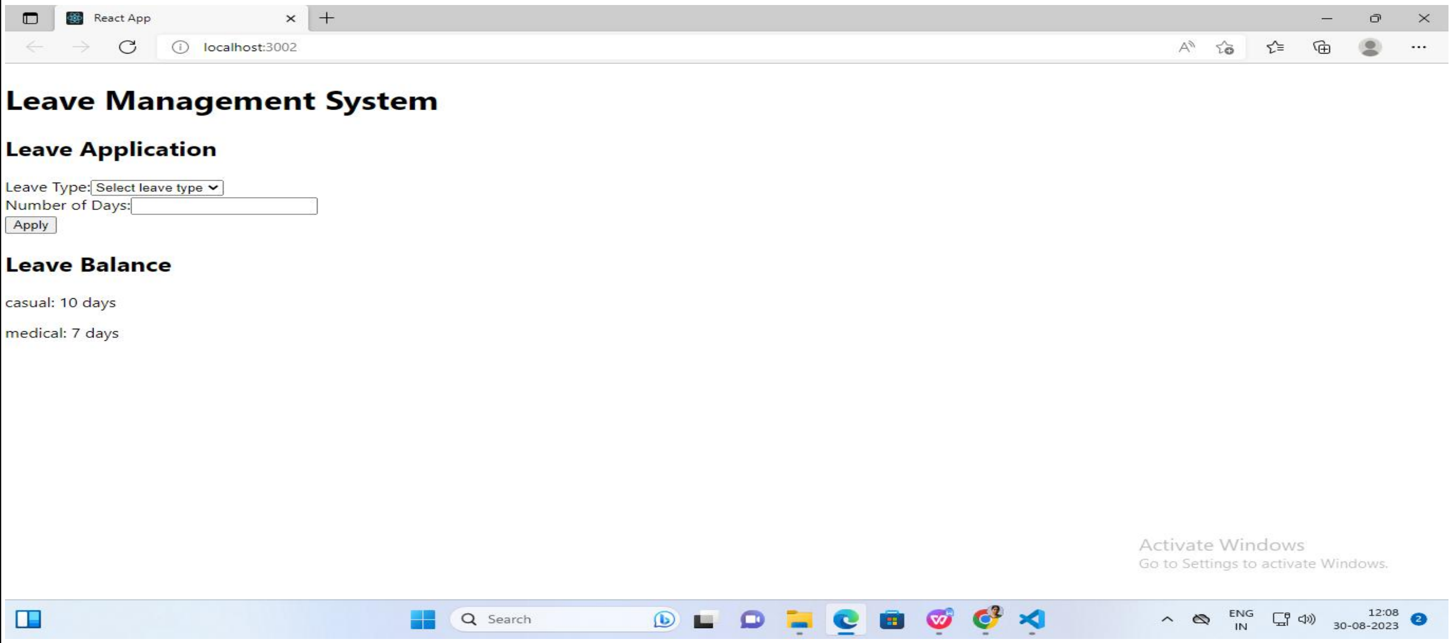
```
import React from 'react';

const LeaveBalance = ({ leaveBalances }) => {
  return (
    <div>
      <h2>Leave Balance</h2>
      <div>
        {Object.keys(leaveBalances).map((leaveType) => (
          <p key={leaveType}>
            {leaveType}: {leaveBalances[leaveType]} days
          </p>
        ))}
      </div>
    </div>
  );
};
```

```
export default LeaveBalance;
```

App.js

```
import React, { useState } from 'react';
import LeaveApplication from './components/LeaveApplication';
import LeaveBalance from './components/LeaveBalance';

const App = () => {
  const [leaveBalances, setLeaveBalances] = useState({
    casual: 10,
    medical: 7,
  });

  const applyLeave = (leaveType, days) => {
    if (leaveBalances[leaveType] >= days) {
      setLeaveBalances((prevBalances) => ({
        ...prevBalances,
        [leaveType]: prevBalances[leaveType] - days,
      }));
      alert('Leave applied successfully!');
    } else {
      alert('Insufficient leave balance.');
    }
  };

  const leaveTypes = Object.keys(leaveBalances);

  return (
    <div>
      <h1>Leave Management System</h1>
      <LeaveApplication leaveTypes={leaveTypes} onApply={applyLeave} />
      <LeaveBalance leaveBalances={leaveBalances} />
    </div>
  );
};

export default App;
```

OUTPUT:



Leave Management System

Leave Application

Leave Type: Select leave type

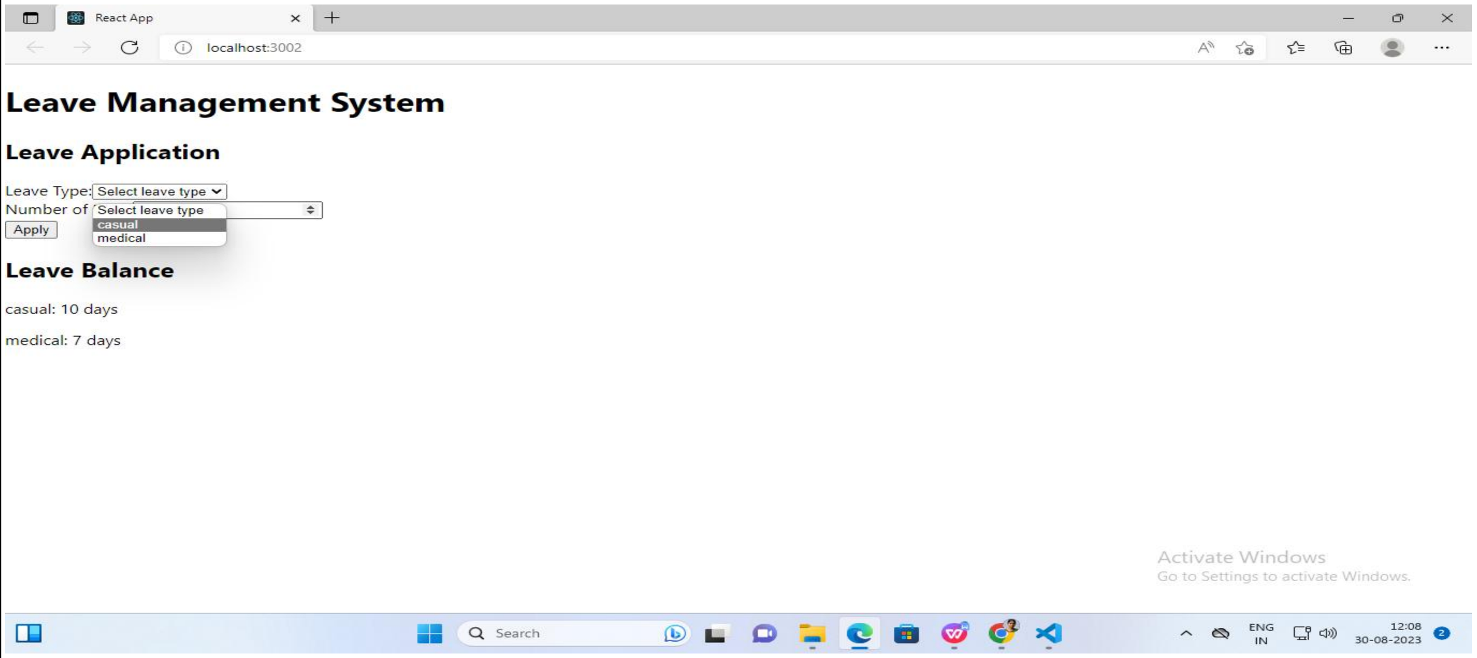
Number of Days:

Apply

Leave Balance

casual: 10 days

medical: 7 days



Leave Management System

Leave Application

Leave Type: Select leave type

Number of

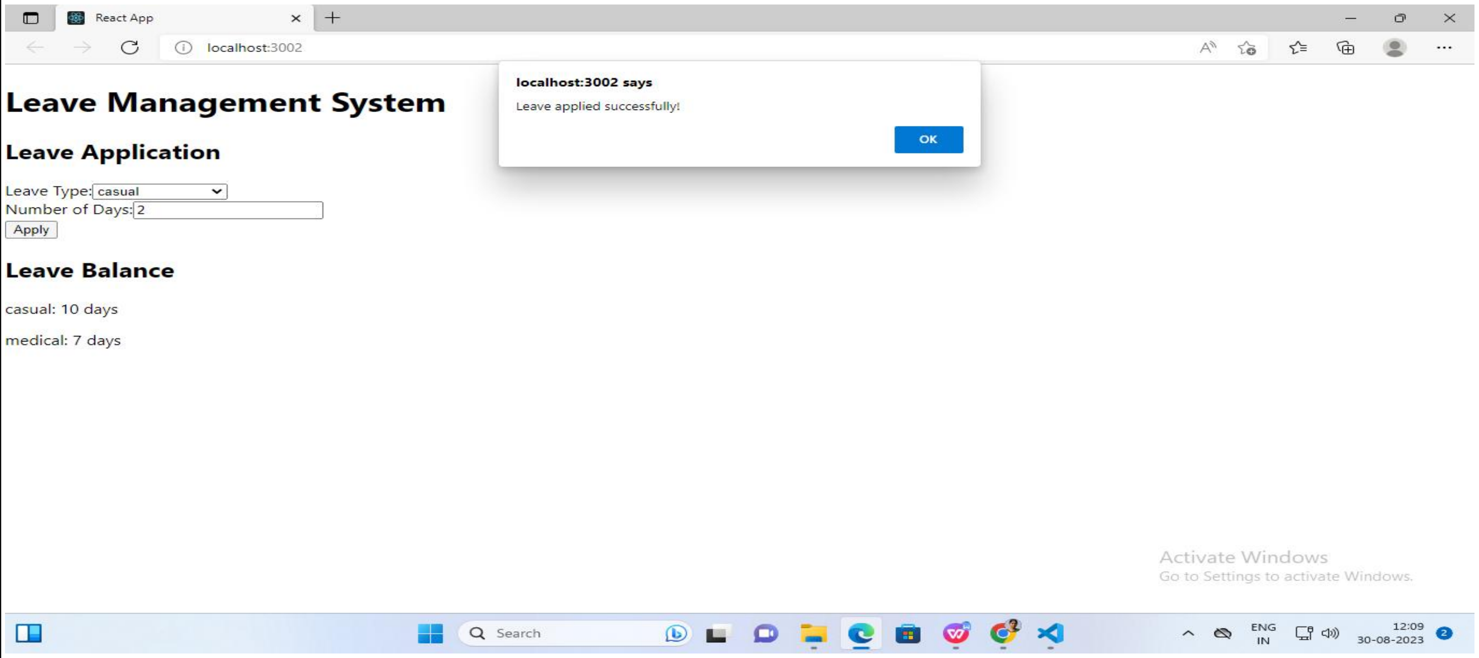
Select leave type
casual
medical

Apply

Leave Balance

casual: 10 days

medical: 7 days



localhost:3002 says

Leave applied successfully!

OK

Leave Management System

Leave Application

Leave Type: casual

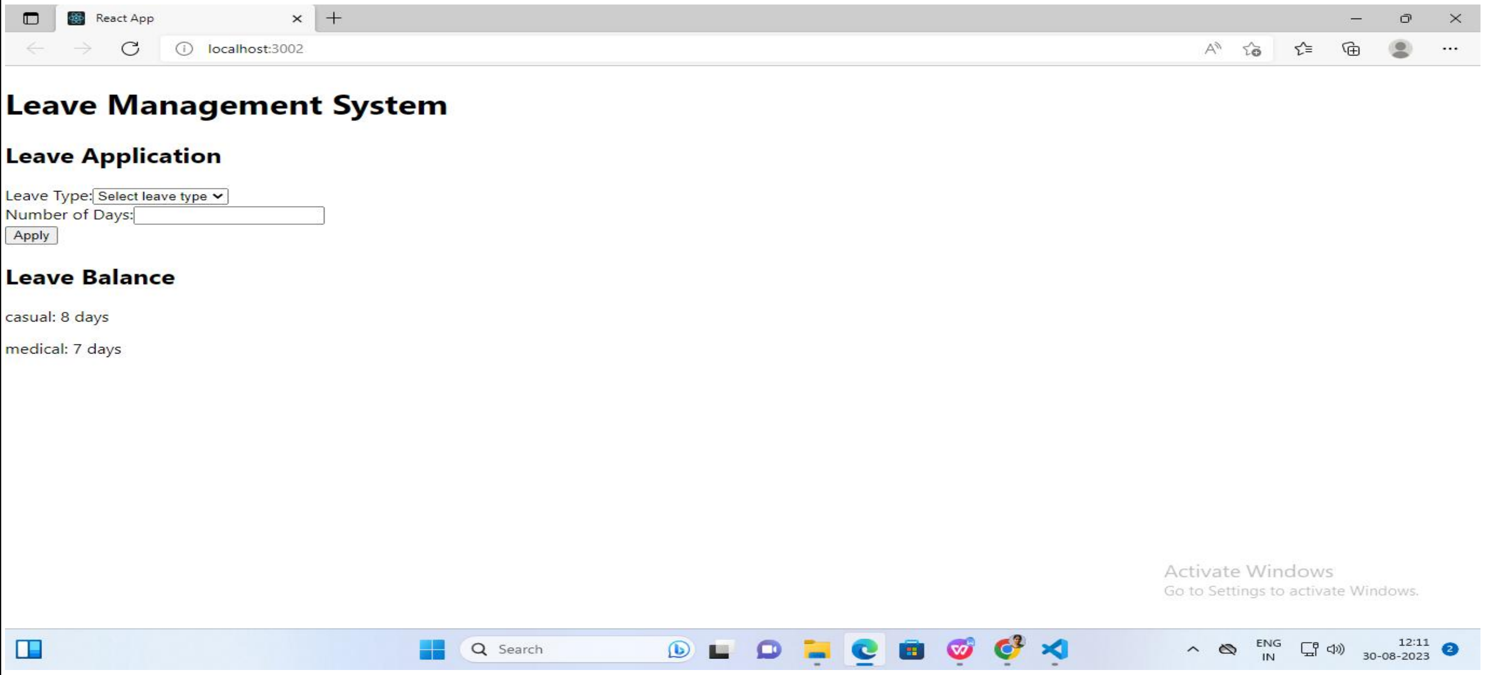
Number of Days: 2

Apply

Leave Balance

casual: 10 days

medical: 7 days



Leave Management System

Leave Application

Leave Type: Select leave type

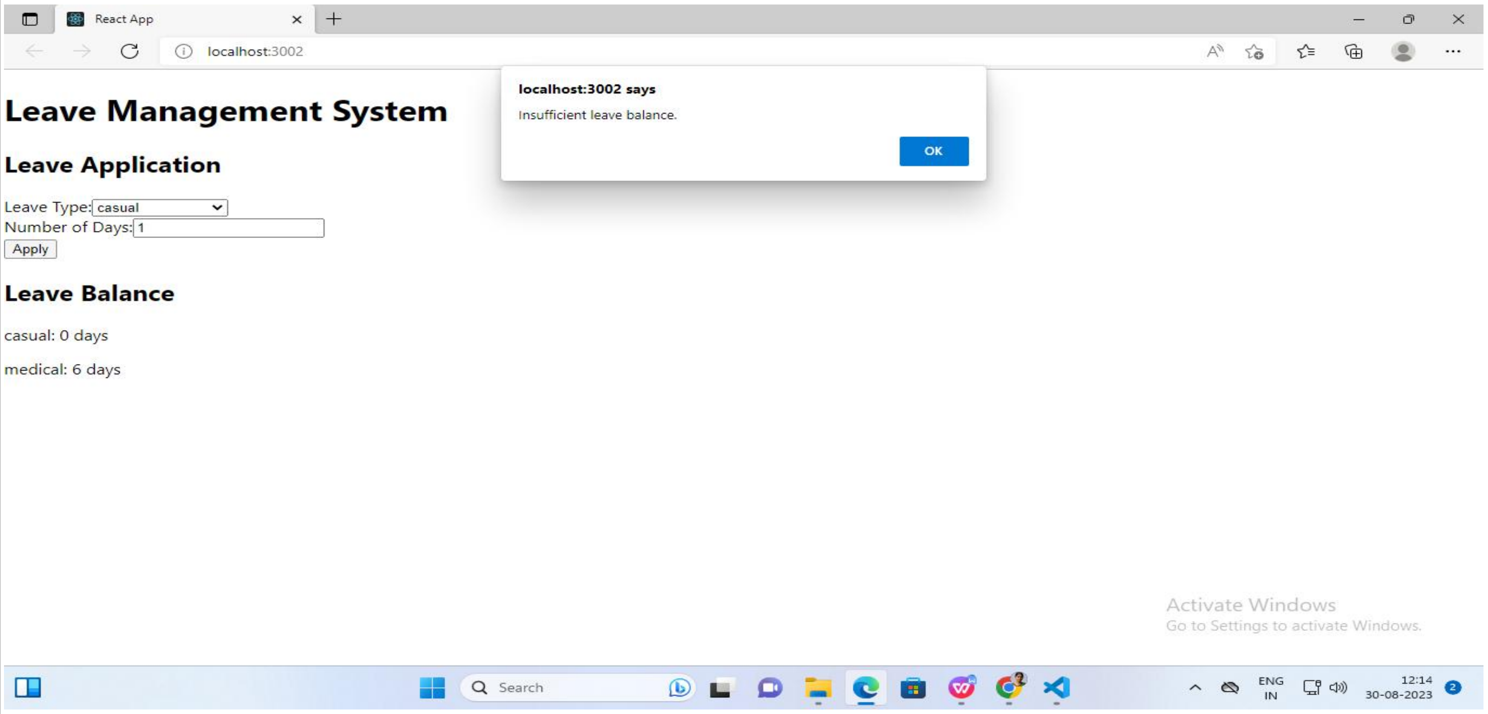
Number of Days:

Apply

Leave Balance

casual: 8 days

medical: 7 days


React App

localhost:3002

localhost:3002 says

Insufficient leave balance.

OK

Leave Management System

Leave Application

Leave Type: casual

Number of Days: 1

Apply

Leave Balance

casual: 0 days

medical: 6 days

RESULT:

Thus the Development of leave management system for an organization was successfully done and executed.

Ex No : 7 DATE :	**Develop a simple dashboard for project management where the statuses of various tasks are available. New tasks can be added and the status of existing tasks can be changed among Pending, InProgress or Completed.**

AIM:

To develop a simple dashboard for project management where the statuses of various tasks are available and new tasks can be added and the status of existing tasks can be changed among Pending, InProgress or completed status will be displayed using ReactJS.

ALGORITHM:

Step 1: Setup
- Create a new React app using create-react-app.

Step 2: Component Creation
- Inside the src folder, create a components directory.
- For each section of our dashboard(TaskForm,TaskList), create a separate .js file inside the components directory.

Step 3: Component Content
- Define the content for each component using JSX.
- Add details about skills, experiences, projects, and contact information.

Step 4: Styling
- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components
- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing
- Run the development server (npm start) to see dashboard website in the browser.

PROGRAM:

TaskForm.js
```
import React, { useState } from 'react';

const TaskForm = ({ onAddTask }) => {
  const [task, setTask] = useState('');
  const [project, setProject] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
    if (task.trim() !== '' && project.trim() !== '') {
      onAddTask(task, project);
      setTask('');
      setProject('');
    }
  };

  return (
    <div>
      <h2>Add New Task</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          value={task}
          onChange={(e) => setTask(e.target.value)}
          placeholder="Enter new task..."
        />
        <input
          type="text"
          value={project}
          onChange={(e) => setProject(e.target.value)}
          placeholder="Enter project name..."
        />
        <button type="submit">Add Task</button>
      </form>
    </div>
  );
};

export default TaskForm;
```

<u>TaskList.js</u>

```
import React from 'react';

const TaskList = ({ tasks, onStatusChange }) => {
  return (
    <div>
      <h2>Task List</h2>
      <ul>
        {tasks.map((task, index) => (
          <li key={index} className={task.status.toLowerCase()} onClick={() => onStatusChange(index)}>
            {task.description} - {task.project} ({task.status})
          </li>
        ))}
      </ul>
    </div>
  );
};
export default TaskList;
```

App.js

```
import React, { useState } from 'react';
import './App.css';
import TaskForm from './Components/TaskForm';
import TaskList from './Components/TaskList';

const App = () => {
  const [tasks, setTasks] = useState([]);

  const addTask = (description, project) => {
    setTasks([...tasks, { description, project, status: 'Pending' }]);
  };

  const changeStatus = (index) => {
    const newTasks = [...tasks];
    const task = newTasks[index];

    switch (task.status) {
      case 'Pending':
        task.status = 'InProgress';
        break;
      case 'InProgress':
        task.status = 'Completed';
        break;
      case 'Completed':
        task.status = 'Pending';
        break;
      default:
        break;
    }

    setTasks(newTasks);
  };

  return (
    <div className="App">
      <h1>Task Management System</h1>
      <TaskForm onAddTask={addTask} />
      <TaskList tasks={tasks} onStatusChange={changeStatus} />
    </div>
  );
};

export default App;
```

App.css

```
.App {
  text-align: center;
  margin: 20px;
}

h1 {
  color: #007bff;
}

form {
  margin-bottom: 20px;
}

button {
  margin-top: 10px;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  margin: 5px;
  display: flex;
  justify-content: space-between;
  background-color: #f0f0f0;
  padding: 5px;
  cursor: pointer;
}

.completed {
  background-color: #b3ffb3;
}

.in-progress {
  background-color: #ffffb3;
```

RESULT:

Thus, the Development of simple dashboard for project management was successfully done and executed.

Ex No : 8 DATE :	**Develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions.**

AIM:

To develop an online survey application where a collection of questions is available and users are asked to answer any random 5 questions using ReactJS.

ALGORITHM:

Step 1: Setup

- Create a new React app using create-react-app.

Step 2: Component Creation

- Inside the src folder, create a components directory.
- For each section of online survey application (Question), create a separate .js file inside the components directory.

Step 3: Component Content

- Define the content for each component using JSX.
- Add details about skills, experiences, projects, and contact information.

Step 4: Styling

- Create corresponding .css files for each component or use a CSS-in-JS approach.
- Apply styling to achieve the desired visual appearance for each section.

Step 5: Integrate Components

- In the App.js file, import the components you've created.
- Render each component within the return statement of the App component.

Step 6:Testing

- Run the development server (npm start) to see online survey website in the browser.

PROGRAM:

Question.js

```
import React from 'react';

const Question = ({ question, onAnswer }) => {
  return (
    <div>
      <h3>{question.text}</h3>
      <div>
        {question.options.map((option, index) => (
          <div key={index}>
            <input
              type="radio"
              id={option}
              name={question.id}
              value={option}
              onChange={() => onAnswer(question.id, option)}
          />
          <label htmlFor={option}>{option}</label>
        </div>
      ))}
    </div>
  </div>
 );
};

export default Question;
```

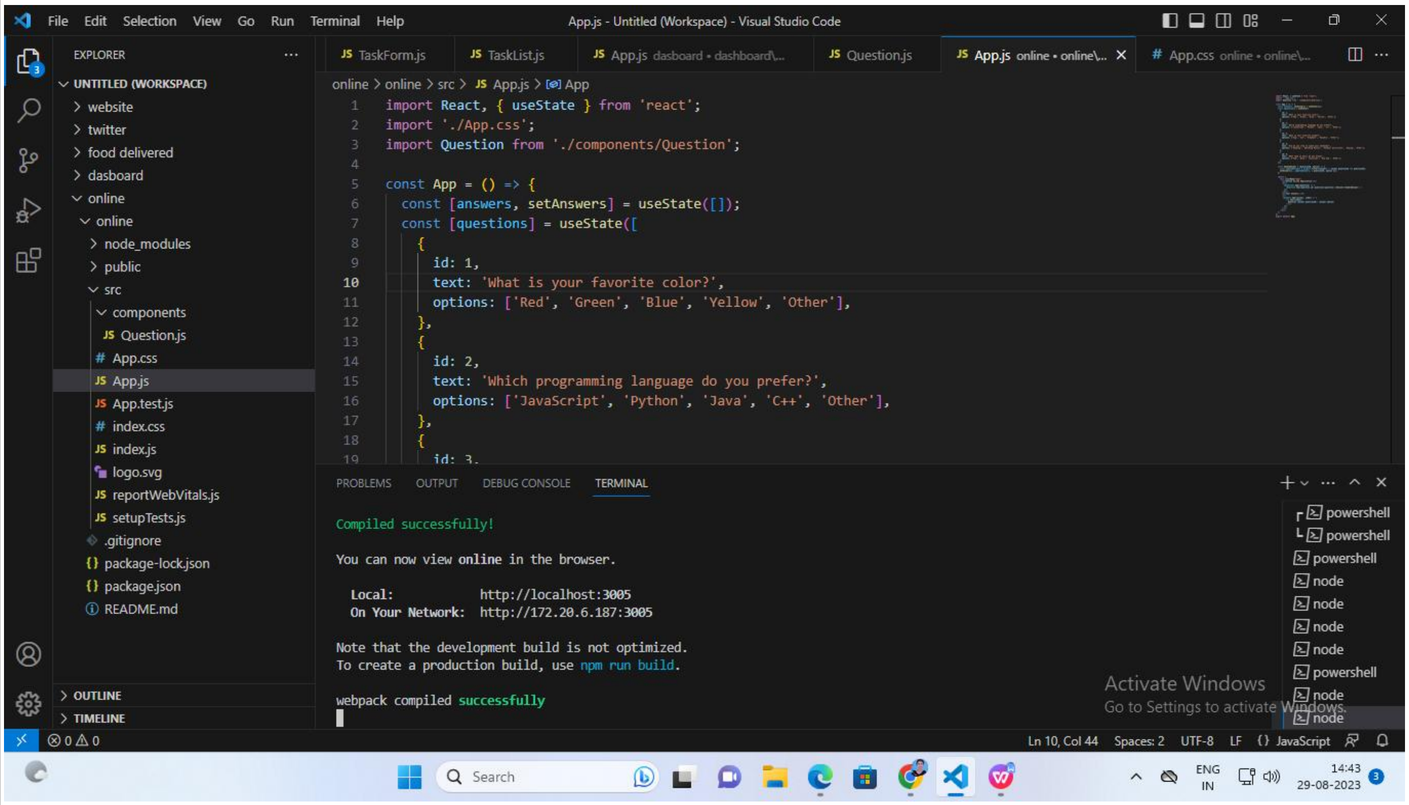
App.js

```
import React, { useState } from 'react';
import './App.css';
import Question from './Components/Question';

const App = () => {
  const [answers, setAnswers] = useState([]);
  const [questions] = useState([
    {
      id: 1,
      text: 'What is your favorite color?',
      options: ['Red', 'Green', 'Blue', 'Yellow', 'Other'],
    },
    {
      id: 2,
      text: 'Which programming language do you prefer?',
      options: ['JavaScript', 'Python', 'Java', 'C++', 'Other'],
    },
    {
      id: 3,
      text: 'What is your favorite animal?',
      options: ['Dog', 'Cat', 'Elephant', 'Dolphin', 'Other'],
    },
    {
      id: 4,
      text: 'How do you like to spend your weekends?',
      options: ['Reading', 'Watching Movies', 'Outdoor Activities', 'Gaming', 'Other'],
    },
    {
      id: 5,
      text: 'What type of music do you enjoy?',
      options: ['Pop', 'Rock', 'Classical', 'Hip Hop', 'Other'],
    },
  ]);

  const handleAnswer = (questionId, option) => {
    const updatedAnswers = answers.filter((answer) => answer.questionId !== questionId);
    setAnswers([...updatedAnswers, { questionId, option }]);
  };

  return (
    <div className="App">
      <h1>Online Survey Application</h1>
      <div>
        {questions.map((question) => (
          <Question key={question.id} question={question} onAnswer={handleAnswer} />
        ))}
      </div>
      <h2>Your answers:</h2>
      <ul>
       {answers.map((answer, index) => (
          <li key={index}>
            Question {answer.questionId}: {answer.option}
          </li>
        ))}
      </ul>
    </div>
  );
};
export default App;
```

App.css

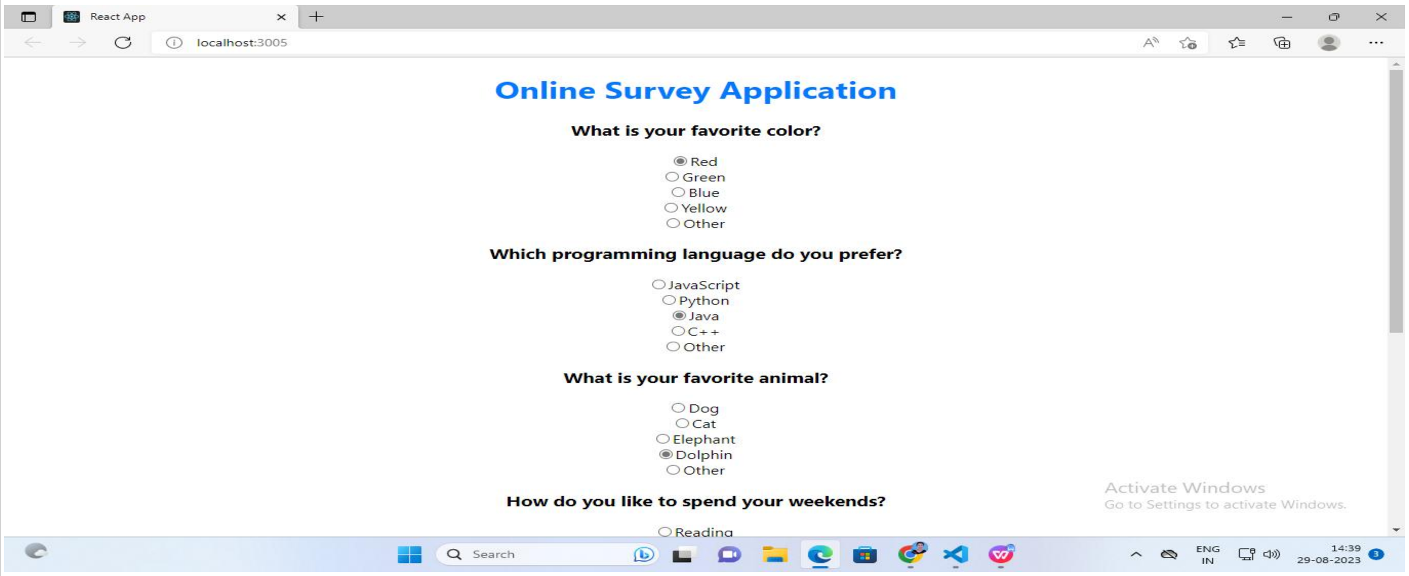
```
.App {
  text-align: center;
  margin: 20px;
}
h1 {
  color: #007bff;
}
ul {
  list-style-type: none;
  padding: 0;
}
li {
  margin: 5px;
}

label {
  margin-right: 5px;
}
```

OUTPUT:



```
import React, { useState } from 'react';
import './App.css';
import Question from './components/Question';

const App = () => {
  const [answers, setAnswers] = useState([]);
  const [questions] = useState([
    {
      id: 1,
      text: 'What is your favorite color?',
      options: ['Red', 'Green', 'Blue', 'Yellow', 'Other'],
    },
    {
      id: 2,
      text: 'Which programming language do you prefer?',
      options: ['JavaScript', 'Python', 'Java', 'C++', 'Other'],
    },
    {
      id: 3,
```

```
Compiled successfully!

You can now view online in the browser.

  Local:            http://localhost:3005
  On Your Network:  http://172.20.6.187:3005

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



Online Survey Application

What is your favorite color?

[✓ Red]
Green
Blue
Yellow
Other

Which programming language do you prefer?

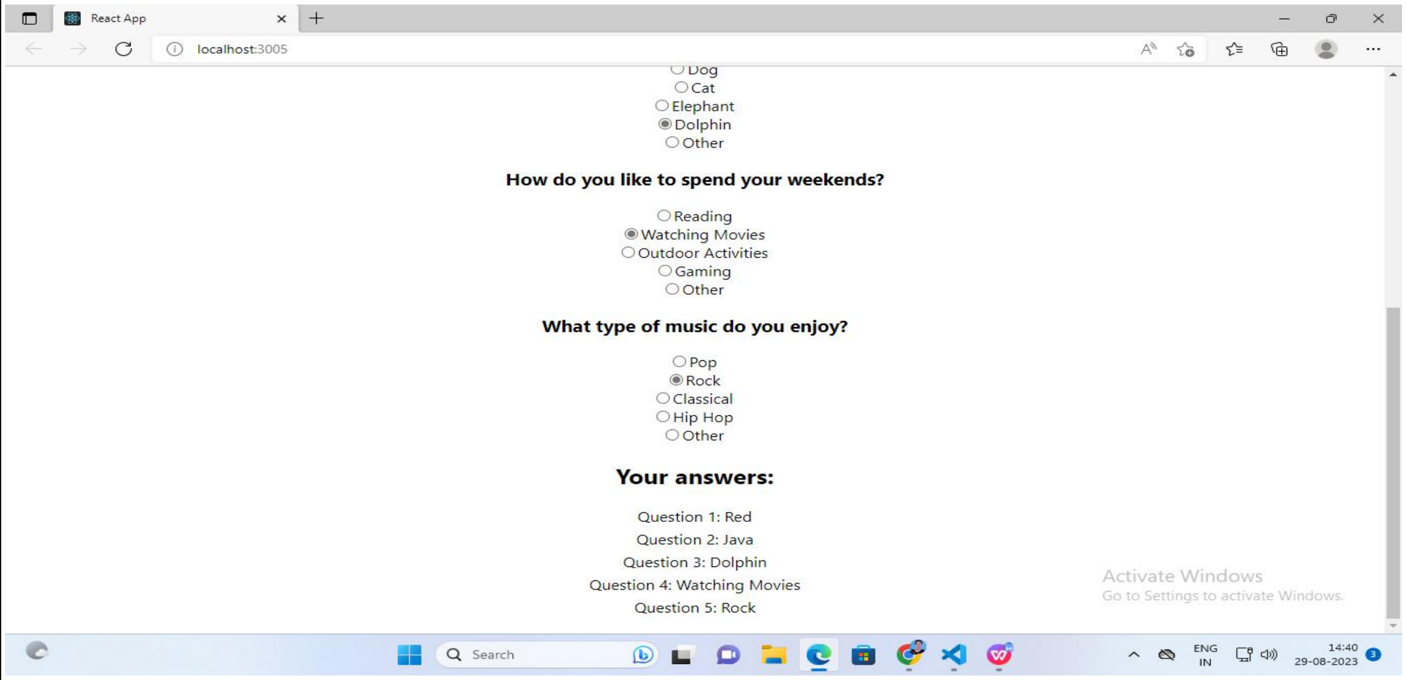
JavaScript
Python
[✓ Java]
C++
Other

What is your favorite animal?

Dog
Cat
Elephant
[✓ Dolphin]
Other

How do you like to spend your weekends?

Reading



React App

localhost:3005

```
○ Dog
○ Cat
○ Elephant
● Dolphin
○ Other

How do you like to spend your weekends?

○ Reading
● Watching Movies
○ Outdoor Activities
○ Gaming
○ Other

What type of music do you enjoy?

○ Pop
● Rock
○ Classical
○ Hip Hop
○ Other

Your answers:

Question 1: Red
Question 2: Java
Question 3: Dolphin
Question 4: Watching Movies
Question 5: Rock
```

RESULT:

Thus, the Development of Online Survey application was successfully done and executed.