Objectives

- **Explain React components**

  React components are the building blocks of a React application. They are **reusable**, **self-contained pieces** of UI that manage their own structure, behavior, and state. You can think of them as custom HTML elements with logic attached.

  A component can be:

  - A simple element like a button
  - A full-featured form
  - Even an entire page

- **Identify the differences between components and JavaScript functions**

Feature	React Component	JavaScript Function
Purpose	Returns JSX (UI element)	Performs logic or calculation
Naming Convention	PascalCase (e.g., MyComponent)	camelCase (e.g., myFunction)
Usage	Used inside JSX	Called like normal functions
Returns	JSX	Data/value

- **Identify the types of components**

1. **Class Components**
    - Defined as ES6 classes
    - Can use lifecycle methods
    - Supports this.state and this.props
2. **Function Components**

- Plain JavaScript functions
- Simpler and leaner
- Can use Hooks (e.g. useState, useEffect)

- **Explain class component**

```
class MyComponent extends React.Component {
 constructor(props) {
  super(props);
  this.state = { count: 0 };
 }
```

```
 render() {
  return <h1>Count: {this.state.count}</h1>;
 }
}
```

Key features:

- Inherits from React.Component
- Defines a constructor to set initial state
- Implements a render() method to describe UI

- **Explain function component**

  Function components are more modern and concise:

```
function MyComponent(props) {
 const [count, setCount] = React.useState(0);

 return <h1>Count: {count}</h1>;
}
```

  Key features:

  - No class syntax required
  - Uses hooks for state and side effects
  - Easier to read and test

- **Define component constructor**

  In class components, the constructor() method:

  - Initializes state
  - Binds event handler methods
  - Receives props as a parameter

  Example:

```
constructor(props) {
 super(props);
 this.state = { key: value };
}
```

- **Define render() function**

  The render() method is **mandatory** in class components. It:

  - Describes what the UI should look like
  - Returns JSX

  Example:

  render() {

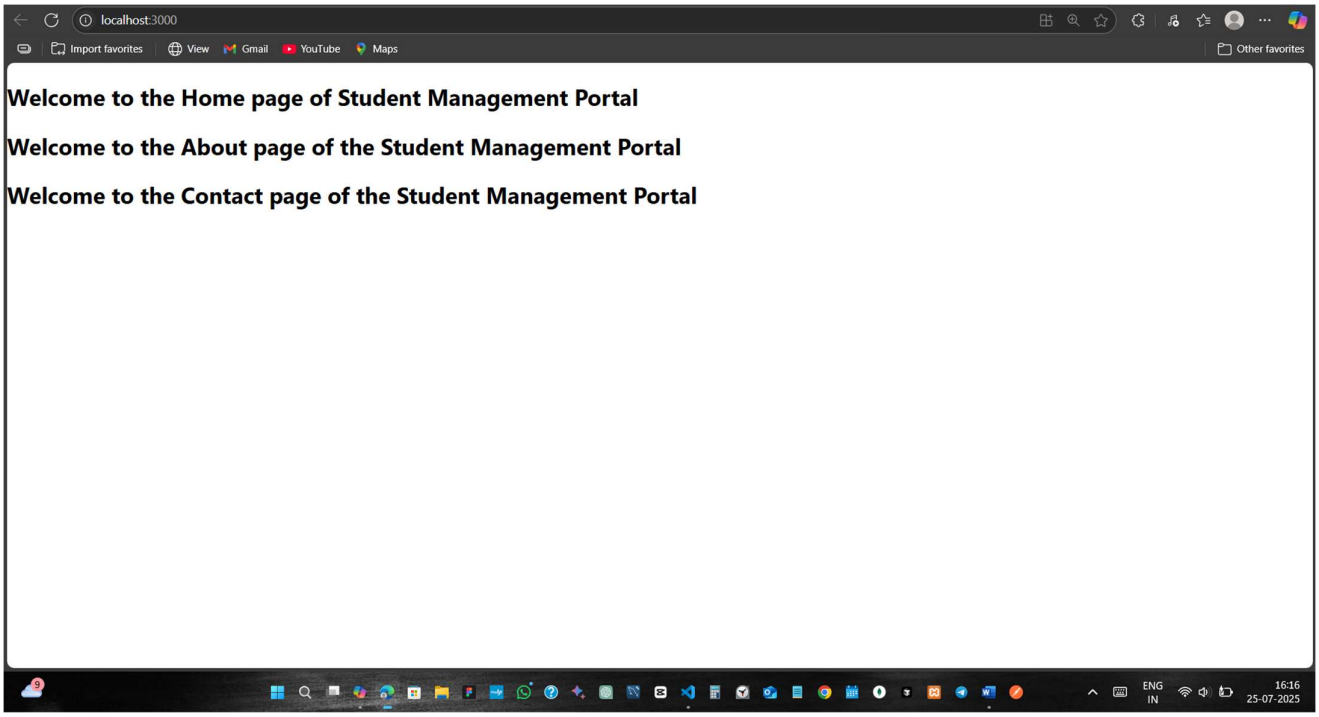
  return Hello, {this.state.name} ;

  }

In this hands-on lab, you will learn how to:

- Create a class component
- Create multiple components
- Render a component

Result



Prerequisites

The following is required to complete this hands-on lab:

- Node.js
- NPM

- Visual Studio Code

Notes

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

1. Create a React project named “StudentApp” type the following command in terminal of Visual studio:

```
C:>npx create-react-app StudentApp
```

2. Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”

3. Type the following code in Home.js

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

import class Home extends Component{
   render(){
      <div>
      <h3> Welcome to the Home Page of Student Management Portal </h3>
      </div>
   }
}
```

4. Under Src folder add another file named “About.js”

5. Repeat the same steps for Creating “About” and “Contact” component by adding a new file as ”About.js”, “Contact.js” under “Src” folder and edit the code as mentioned for “Home” Component.

6. Edit the App.js to invoke the Home, About and Contact component as follows:

```
import logo from './logo.svg';
import './App.css';
import {Home} from './Components/Home';
import {About} from './Components/About';
import {Contact} from './Components/Contact';

function App() {
  return (
    <div className="container">
     <Home/>
     <About/>
     <Contact/>
    </div>
  );
}

export default App;
```

7. In command Prompt, navigate into StudentApp and execute the code by typing the following command:

```
C:\studentapp>npm start
```

8. Open browser and type “localhost:3000” in the address bar: