1. ReactJS-HOL

1. Define SPA and its benefits

SPA (Single Page Application) is a web application or website that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from the server. This approach makes the application faster and provides a smoother user experience.

Benefits of SPA:

- Faster Navigation: No need to reload the whole page.
- Improved User Experience: Feels more like a desktop application.
- Less Server Load: Only data is exchanged, not full pages.
- Easier Debugging: Most SPAs use browser developer tools effectively.

2. Define React and identify its working

React is an open-source JavaScript library developed by Facebook for building user interfaces, especially for single-page applications. It allows developers to create large web applications that can update and render efficiently with changing data.

How React works:

- React uses components, which are reusable pieces of UI.
- It creates a virtual DOM (a copy of the real DOM) and updates only the parts that change.
- React follows a unidirectional data flow, making the code easier to understand and debug.

3. Identify the differences between SPA and MPA

Feature	SPA (Single Page Application)	MPA (Multi Page Application)
Page Load	Loads once and updates dynamically	Loads a new page for each request
Speed	Faster after initial load	Slower due to full-page reloads
Development	Mostly uses JavaScript frameworks	Often uses server-side rendering
Navigation	Uses JavaScript routing	Traditional server-side routing
SEO	Harder to optimize	Easier to manage

4. Explain Pros & Cons of Single-Page Application

Pros:

- Fast and responsive experience
- Reduces server traffic
- Easier to develop and test once structure is set
- Better caching possibilities

Cons:

- SEO optimization is harder
- May have slower initial loading time
- Can be complex for large apps
- JavaScript must be enabled in the browser

5. Explain about React

React is a component-based JavaScript library used to build interactive user interfaces. It helps in building reusable UI components, manages the application's view layer, and efficiently updates the UI when the data changes. It supports both client-side and server-side rendering and is widely used for developing SPAs.

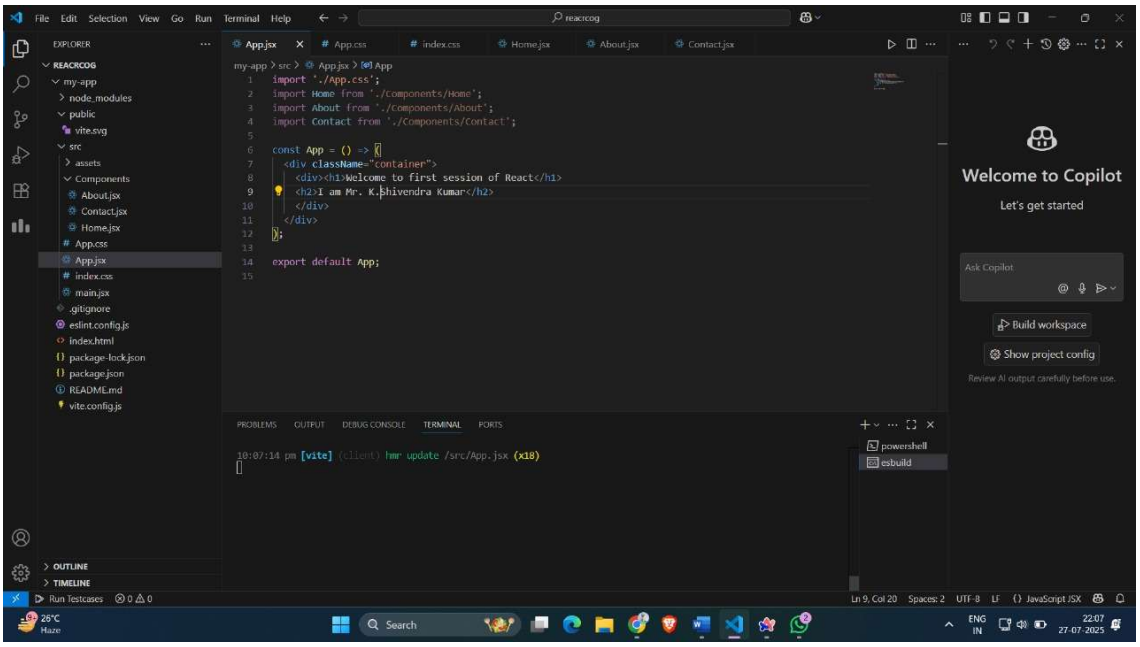
6. Define Virtual DOM

The Virtual DOM is a lightweight copy of the actual DOM used by React. Whenever the application state changes, React creates a new virtual DOM and compares it with the previous one using a process called diffing. It then updates only the changed elements in the real DOM. This makes the application faster and more efficient.

7. Explain Features of React

- Component-Based Architecture: UI is divided into reusable components.
- Virtual DOM: Speeds up rendering and improves performance.
- JSX Syntax: Allows writing HTML inside JavaScript.
- One-Way Data Binding: Makes data flow predictable.
- Declarative UI: Easy to design interactive interfaces.
- Strong Community Support: React has a large and active community.
- Fast Rendering: Only the necessary parts of the UI are re-rendered.

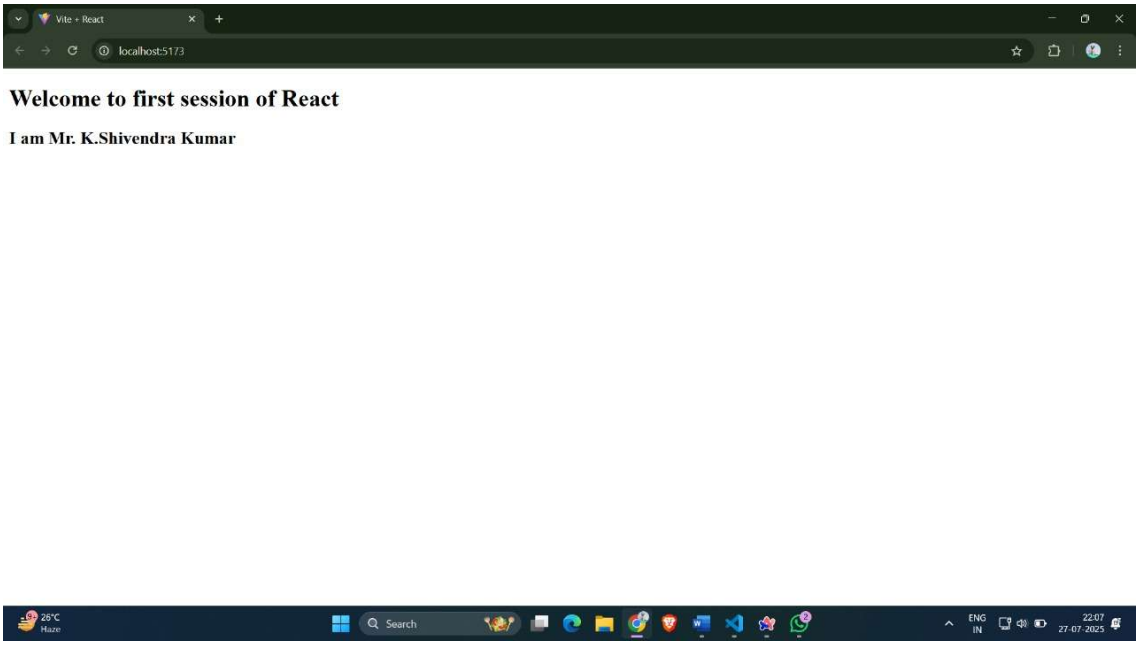
Code:


File Edit Selection View Go Run Terminal Help
reacrcog
EXPLORER
REACRCOG
my-app
node_modules
public
vite.svg
src
assets
Components
About.jsx
Contact.jsx
Home.jsx
App.css
App.jsx
index.css
main.jsx
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
README.md
vite.config.js
App.jsx
App.css
index.css
Home.jsx
About.jsx
Contact.jsx
my-app > src > App.jsx > App
import './App.css';
import Home from './Components/Home';
import About from './Components/About';
import Contact from './Components/Contact';
const App = () => {

<h1>Welcome to first session of React</h1>
<h2>I am Mr. K.Shivendra Kumar</h2>

};
export default App;
Welcome to Copilot
Let's get started
Ask Copilot
Build workspace
Show project config
Review AI output carefully before use.
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
10:07:14 pm [vite] (client) hmr update /src/App.jsx (x18)
powershell
esbuild
OUTLINE
TIMELINE
Run Testcases
Ln 9, Col 20
Spaces: 2
UTF-8
LF
JavaScript JSX
26°C
Haze
Search
ENG
IN
22:07
27-07-2025

Output:


Vite + React
localhost:5173
Welcome to first session of React
I am Mr. K.Shivendra Kumar
26°C
Haze
Search
ENG
IN
22:07
27-07-2025

2. ReactJS-HOL

1. Explain React Components

React components are the building blocks of any React application. They are independent, reusable pieces of UI that return elements to be displayed on the screen. Components allow developers to break down the user interface into smaller, manageable parts.

2. Identify the Differences Between Components and JavaScript Functions

Feature	React Components	JavaScript Functions
Purpose	Used to create UI elements	Used to perform logic or operations
Return Type	Returns JSX (HTML-like code)	Returns values or executes logic
Reusability	Designed to be reused in UI	May or may not be reused
Lifecycle Methods	Have access to React lifecycle (class)	No built-in lifecycle methods
State Management	Can manage state (class/function with hooks)	Cannot manage state on their own

3. Identify the Types of Components

React has mainly two types of components:

1. Class Components – Components that are ES6 classes and extend React.Component.
2. Function Components – Simple JavaScript functions that return JSX.

4. Explain Class Component

A Class Component is a type of React component defined using a JavaScript class that extends React.Component. It can hold its own state and use lifecycle methods such as componentDidMount, componentDidUpdate, etc.

```
class Home extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the Home page of Student Management portal</h3>
```

```
    </div>
  );
 }
}
```

5. Explain Function Component

A Function Component is a simpler way to write components. It is a regular JavaScript function that takes props as an argument and returns JSX. With the introduction of Hooks, function components can also manage state and side effects.

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

6. Define Component Constructor

In class components, the constructor() is a special method used to initialize state and bind methods. It is called before the component is mounted.

```
constructor(props) {
 super(props);
 this.state = { name: "React" };
}
```

7. Define render() Function

The render() function is a required method in class components. It describes what should be displayed on the screen. It returns the JSX code to be rendered in the browser.

```
render() {
  return (
   <div>
    <h3>Welcome to the Home page of Student Management portal</h3>
   </div>
  ); } }
```

Code:

App.js:

```
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;
```

Home.js:

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

class Home extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the Home page of Student Management portal</h3>
      </div>
    );
  }
}

export default Home;
```

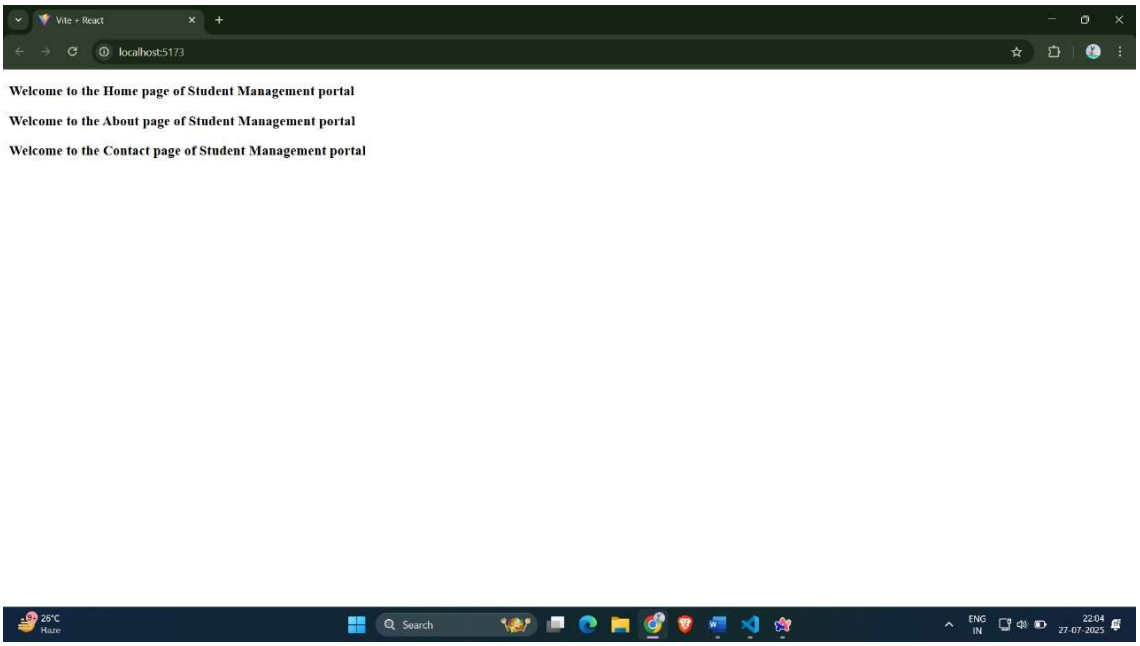
About.js:

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

class About extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the About page of Student Management portal</h3>
      </div>
    );
  }
}

export default About;
```

Contact.js:

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

class Contact extends Component {
  render() {
    return (
      <div>
        <h3>Welcome to the Contact page of Student Management portal</h3>
      </div>
    );
  }
}

export default Contact;
```

Output:


Vite + React
localhost:5173
Welcome to the Home page of Student Management portal
Welcome to the About page of Student Management portal
Welcome to the Contact page of Student Management portal
26°C
Haze
Search
ENG
IN
22:04
27-07-2025

3. ReactJS-HOL

Code:

App.js:

```
import { CalculateScore } from '../src/Components/CalculateScore';

function App() {
  return (
    <div>
      <CalculateScore
        Name={"K. Shivendra KUmar"}
        School={"Sri Chaitanya E-Techno School "}
        total={588}
        goal={6}
      />
    </div>
  )
}

export default App;
```

CalculateScore.js:

```
import '../Stylesheets/mystyle.css';

const percentToDecimal = (decimal) => {
    return (decimal.toFixed(2) + '%')
}

const calcScore = (total, goal) => {
    return percentToDecimal(total/goal)
}

export const CalculateScore = ({ Name, School, total, goal }) => (
    <div className="formatstyle">
        <h1><font color="Brown">Student Details:</font></h1>
        <div className="Name">
            <b><span> Name: </span> </b>
            <span>{Name}</span>
        </div>
```

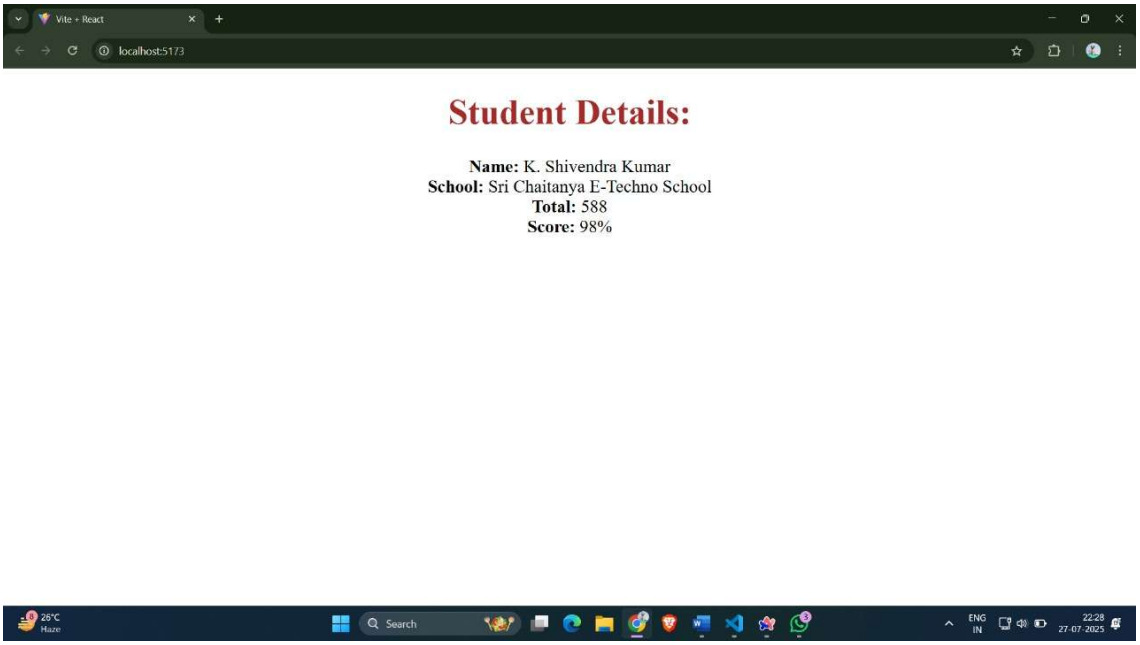
```
      <div className="School">
        <b><span> School: </span> </b>
        <span>{School}</span>
      </div>
      <div className="Total">
        <b><span>Total:</span> </b>
        <span>{total}</span>
        <span>Marks</span>
      </div>
      <div className="Score">
        <b>Score:</b>
        <span>
          {calcScore(
            total,
            goal
          )}
        </span>
      </div>
    </div>
)
```

mystyle.css:

```
.Name {
  font-weight: 300;
  color: blue;
}

.School {
  color: crimson;
}

.Total {
  color: darkmagenta;
}

.formatstyle {
  text-align: center;
  font-size: large;
}

.Score {
```

```
    color: forestgreen;
}
```

Output:



4. ReactJS-HOL

1. Explain the Need and Benefits of Component Life Cycle

The component lifecycle in React defines the series of phases a component goes through from creation to removal. Understanding this lifecycle helps developers control what happens at different stages of a component's existence.

Need for Component Lifecycle:

- To perform specific actions at different stages (e.g., fetching data after rendering).
- To manage side effects like API calls, event listeners, or DOM manipulations.
- To handle performance optimization and resource cleanup.

Benefits:

- Better control over behavior: Helps manage state and props during updates.
- Efficient resource management: Cleanup prevents memory leaks.
- Improved performance: Optimize re-rendering using lifecycle methods.
- Ease in debugging: Lifecycle hooks help track the component's flow.

2. Identify Various Lifecycle Hook Methods

React class components have lifecycle methods grouped into three main phases:

1. Mounting (component is being added to the DOM):

- constructor()
- static getDerivedStateFromProps()
- render()
- componentDidMount()

2. Updating (component is being re-rendered due to state/prop change):

- static getDerivedStateFromProps()
- shouldComponentUpdate()
- render()
- getSnapshotBeforeUpdate()
- componentDidUpdate()

3. Unmounting (component is being removed from the DOM):

- componentWillUnmount()

3. List the Sequence of Steps in Rendering a Component

Here's the sequence of steps when a class component is rendered (Mounting phase):

1. constructor() – Initializes state and binds methods.
2. getDerivedStateFromProps() – Updates state from props before rendering.
3. render() – Returns JSX to be displayed.
4. componentDidMount() – Called after the component is rendered to the DOM.

If the component updates (due to props/state changes), the following sequence occurs:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

When the component is removed:

- componentWillUnmount() is called for cleanup.

Code:

App.js:

```
import React from 'react';
import Posts from './Posts';

function App() {
  return (
    <div className="App">
      <Posts />
    </div>
  );
}

export default App;
```

Post.js:

```
class Post {
    constructor (id, title, body)
    {
        this.id=id;
        this.title=title;
        this.body=body;
    }
}
export default Post;
```

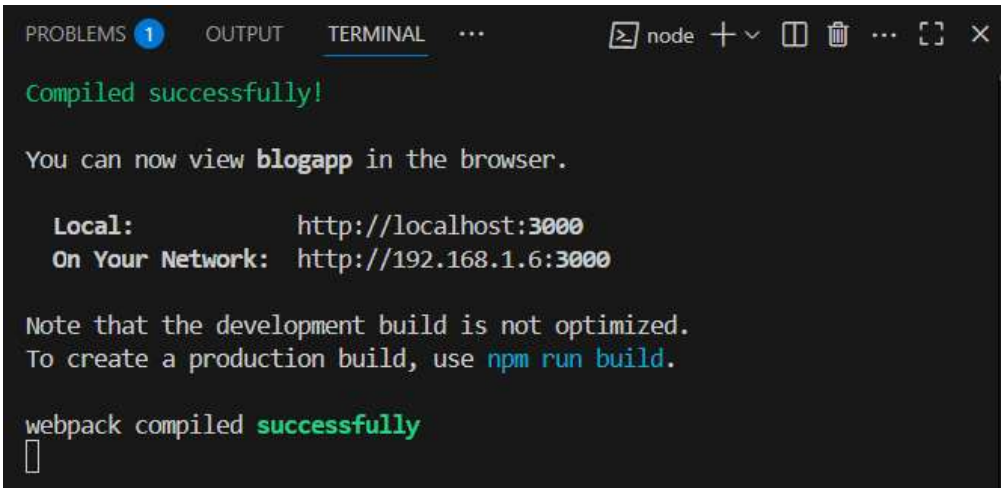
Posts.js:

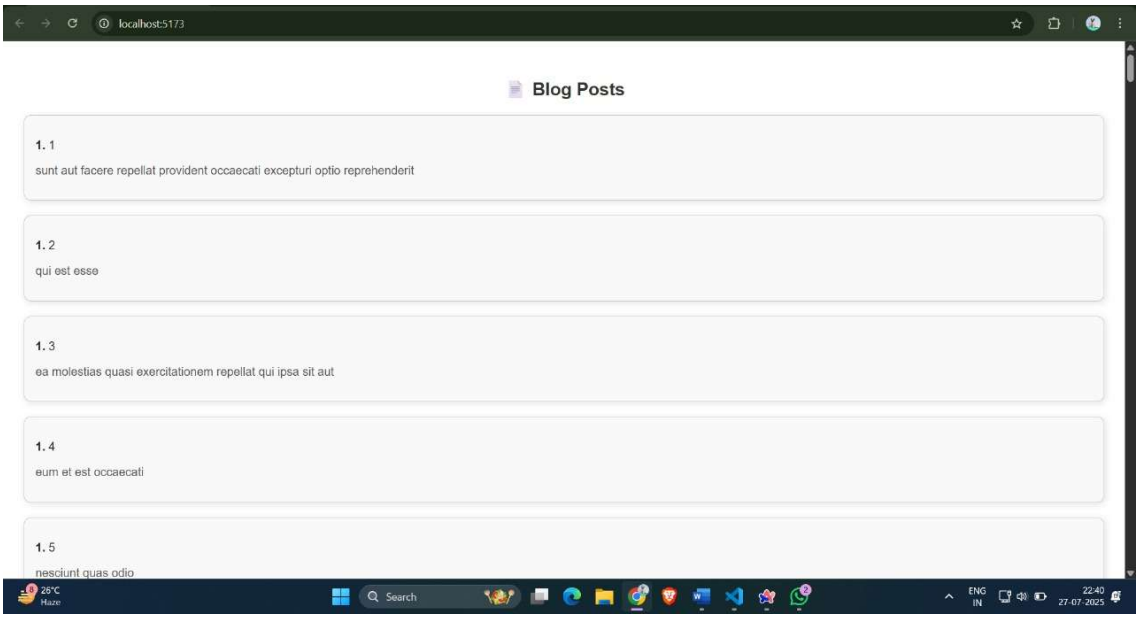
```
import React, { Component } from 'react';
import Post from './Post';

class Posts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      posts: [],
    };
  }

  // Step 6: loadPosts method
```

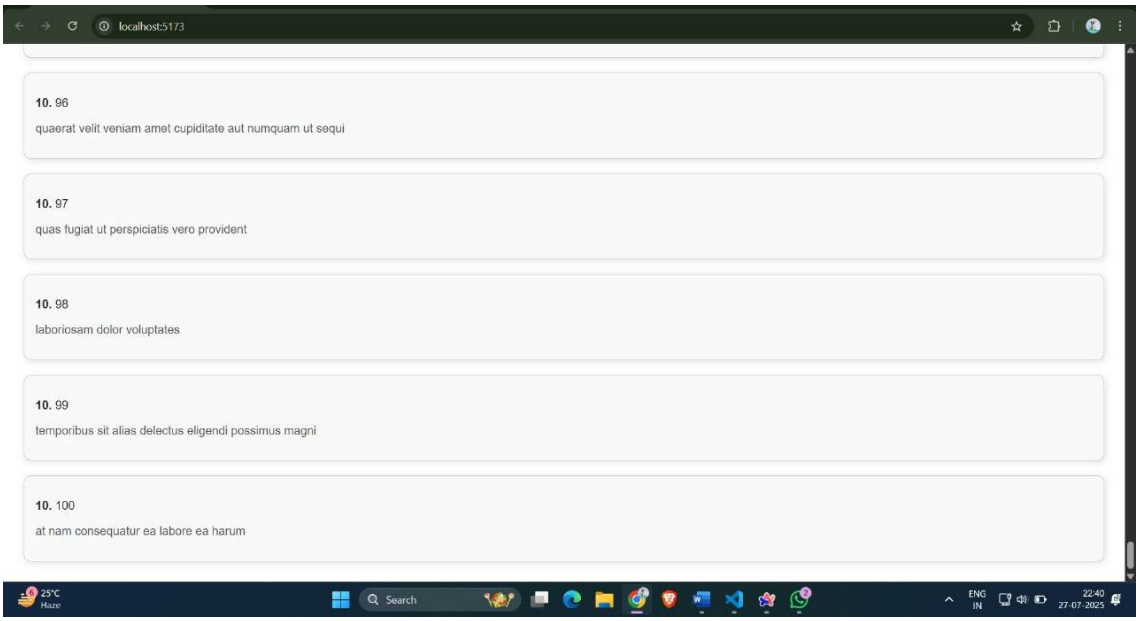
```
loadPosts = () => {
  fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(data => {
      const posts = data.map(p => new Post(p.userId, p.id, p.title, p.body));
      this.setState({ posts });
    })
    .catch(error => {
      console.error("Error fetching posts:", error);
      alert("Error loading posts");
    });
};

// Step 7: componentDidMount
componentDidMount() {
  this.loadPosts();
}

// Step 9: Error handling
componentDidCatch(error, info) {
  alert("Something went wrong! 🤯");
  console.log("Error caught:", error, info);
}

// Step 8: Render Method
render() {
return (
  <div style={{ padding: '20px', fontFamily: 'Arial' }}>
    <h2 style={{ textAlign: 'center', color: '#333' }}>📚 Blog Posts</h2>
    <div style={{ display: 'flex', flexDirection: 'column', gap: '20px' }}>
      {this.state.posts.map(post => (
        <div key={post.id} style={{
          border: '1px solid #ccc',
          padding: '15px',
          borderRadius: '10px',
          boxShadow: '2px 2px 8px rgba(0, 0, 0, 0.1)',
          backgroundColor: '#f9f9f9'
        }}>
          <h3 style={{ color: '#2a2a2a', marginBottom: '10px' }}>
            {post.id}. {post.title}
          </h3>
          <p style={{ color: '#555' }}>{post.body}</p>
        </div>
```

```
      ))}
    </div>
  </div>
 );
}

}

export default Posts;
```

Output:






localhost:5173
10. 96
quaerat velit veniam amet cupiditate aut numquam ut sequi
10. 97
quas fugiat ut perspiciatis vero provident
10. 98
laboriosam dolor voluptates
10. 99
temporibus sit alias delectus eligendi possimus magni
10. 100
at nam consequatur ea labore ea harum
25°C
Haze
Search
ENG
IN
22:40
27-07-2025

5. ReactJS-HOL

Code:

CohortDetails.js:

```
import styles from './CohortDetails.module.css';

function CohortDetails(props) {
   const headingStyle = {
      color: props.cohort.currentStatus === 'ongoing' ? 'green' : 'blue'
   };

   return (
      <div className={styles.box}>
         <h3 style={headingStyle}>
            {props.cohort.cohortCode} -
            <span>{props.cohort.technology}</span>
         </h3>
         <dl>
            <dt>Started On</dt>
            <dd>{props.cohort.startDate}</dd>
            <dt>Current Status</dt>
            <dd>{props.cohort.currentStatus}</dd>
            <dt>Coach</dt>
            <dd>{props.cohort.coachName}</dd>
            <dt>Trainer</dt>
            <dd>{props.cohort.trainerName}</dd>
         </dl>
      </div>
   );
}

export default CohortDetails;
```

CohortDetails.module.css:

```
.box {
  width: 300px;
  display: inline-block;
  margin: 10px;
  padding: 10px 20px;
```

```
  border: 1px solid black;
  border-radius: 10px;
}

dt {
  font-weight: 500;
}
```

Output: