

React Assignment: Lists, Hooks, LocalStorage, and API Project

1. Lists

Theory:

In React, lists are used to display a collection of items such as names, products, or posts.

They are usually created using JavaScript's `.map()` method, which helps loop through an array and return JSX elements.

Each list item should have a unique key prop, which helps React identify which items are changed, added, or removed.

Lists improve UI efficiency and make components reusable and dynamic.

Key Points:

Lists use `.map()` to render multiple elements.

Each child in a list must have a unique key.

Lists can be nested or combined with conditional rendering.

Practical Example:

```
import React from 'react';
```

```
function ListExample() {
```

```
  const names = ['Amir', 'Shyam', 'Vijay'];
```

```
return (  
  <ul>  
    {names.map((name, index) => (  
      <li key={index}>{name}</li>  
    ))}  
  </ul>  
);  
}
```

```
export default ListExample;
```

Output:

A list displaying:

Amir

Shyam

Vijay

2. Hooks

Theory:

Hooks are special functions in React that let you use state and other React features without writing a class.

They were introduced in React 16.8 and made function components more powerful.

Common Hooks:

useState – to manage component state.

useEffect – to handle side effects (API calls, timers).

useContext – to share data across components.

useRef – to access DOM elements directly.

Hooks simplify code and help in writing cleaner, reusable components.

Practical Example:

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

```
function Counter() {
```

```
  const [count, setCount] = useState(0);
```

```
  return (
```

```
    <div>
```

```
      <h2>Count: {count}</h2>
```

```
      <button onClick={() => setCount(count + 1)}>Increment</button>
```

```
    </div>
```

```
  );
```

```
}
```

```
export default Counter;
```

Output:

A counter that increases its value each time the button is clicked.

3. LocalStorage

Theory:

LocalStorage is a Web API that stores data persistently in the user's browser.

The data remains even after refreshing or closing the tab (until manually cleared).

It's commonly used in React for saving:

User preferences (dark/light mode)

Authentication tokens

Form data or settings

You can access LocalStorage with:

```
localStorage.setItem('key', 'value')
```

```
localStorage.getItem('key')
```

```
localStorage.removeItem('key')
```

Practical Example:

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

```
function LocalStorageExample() {
```

```
  const [name, setName] = useState("");
```

```
  useEffect(() => {
```

```
    const savedName = localStorage.getItem('userName');
```

```
    if (savedName) setName(savedName);
```

```
  }, []);
```

```
  const handleSave = () => {
```

```
    localStorage.setItem('userName', name);
```

```
    alert('Name saved!');
```

```
  };
```

```
  return (
```

```
    <div>
```

```
      <input
```

```
        type="text"
```

```
        placeholder="Enter your name"
```

```
    value={name}
    onChange={(e) => setName(e.target.value)}
  />
  <button onClick={handleSave}>Save Name</button>
</div>
);
}
```

```
export default LocalStorageExample;
```

Output:

User enters a name, clicks “Save Name,” and the value is stored in browser LocalStorage.

4. API Project

Theory:

APIs (Application Programming Interfaces) allow applications to communicate with each other.

In React, APIs are used to fetch external data (e.g., users, posts, weather).

You can use `fetch()` or `Axios` to retrieve data from servers and display it dynamically.

API calls are usually made inside the `useEffect()` hook to ensure they run after the component mounts.

Practical Example:

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

```
function ApiProject() {  
  
  const [users, setUsers] = useState([]);  
  
  useEffect(() => {  
  
    fetch('https://jsonplaceholder.typicode.com/users')  
  
      .then(response => response.json())  
  
      .then(data => setUsers(data));  
  
  }, []);  
  
  return (  
  
    <div>  
  
      <h2>User List</h2>  
  
      <ul>  
  
        {users.map(user => (  
  
          <li key={user.id}>{user.name}</li>  
  
        ))}  
  
      </ul>  
  
    </div>  
  
  );  
}  
  
export default ApiProject;
```

Output:

Displays a list of users fetched from the API <https://jsonplaceholder.typicode.com/users>.