Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Practical No. 5
Study and implementation of ReactJs

Problem Statement 0: Basics of ReactJs

1. What is React and what problem does it solve?
   - Answer: React is an open-source JavaScript library developed by Facebook for building user interfaces, especially single-page applications. It simplifies the process of creating interactive UIs by breaking them into reusable components, and it improves performance with its virtual DOM that efficiently updates only the parts of the UI that change.

2. What are React components and how are they used?
   - Answer: React components are the building blocks of a React application. They are self-contained, reusable pieces of code that represent parts of the user interface. Components can be defined as either class-based or functional (using functions), and they allow developers to break down complex UIs into simpler, manageable parts.

3. What is JSX in React?
   - Answer: JSX (JavaScript XML) is a syntax extension for JavaScript that allows developers to write HTML-like code directly within JavaScript. It makes the code more readable and helps in describing the UI structure in a declarative way. Under the hood, JSX is compiled into regular JavaScript function calls.

4. What are props in React and how do they differ from state?
   - Answer: Props (short for properties) are read-only data passed from a parent component to a child component. They allow for component customization and reusability. In contrast, state is a mutable data structure managed within a component. While props are used to pass data down the component tree, state is used to store data that can change over time within the component and trigger re-renders when updated.

5. What is state in React and how does it work?
   - Answer: State is an object that holds data that may change over the lifecycle of a component. It is managed within the component (using methods like setState in class components or the useState hook in functional components) and is used to control the component's behavior and rendering. When state changes, React re-renders the component to reflect the new data.

6. What are React lifecycle methods, and why are they important?
- Answer: React lifecycle methods are special methods in class components (and similar hooks in functional components) that are called at different stages of a component's life, such as mounting, updating, and unmounting. They allow developers to perform actions like fetching data, setting up subscriptions, or cleaning up resources at appropriate times, thereby managing side effects and optimizing performance.

7. Elaborate on the following with respect to ReactJs:
a. Event Handling
- Answer: React uses synthetic events which wrap native events to ensure cross-browser compatibility. Event handlers in React are passed as props to elements and are typically written as functions. They enable developers to respond to user interactions such as clicks, form submissions, or key presses.

b. Conditional Rendering
- Answer: Conditional rendering in React allows components to render different outputs based on certain conditions. This can be done using JavaScript conditional statements like if/else or ternary operators within the JSX, ensuring that the UI dynamically reflects the state or props of the component.

c. Lists and Keys
- Answer: When rendering lists of elements, React requires a unique key for each element to help identify which items have changed, been added, or removed. This improves the efficiency of updates and re-renders by enabling React to track elements in the list more accurately.

d. Forms
- Answer: React handles forms using controlled components where the form data is handled by the component's state. This approach provides full control over the data and makes it easier to validate user input, handle submissions, and manage form state.

e. Hooks
- Answer: Hooks are functions that let developers use state and other React features in functional components. Common hooks include useState for state management, useEffect for handling side effects, useContext for accessing context, and many others that simplify component logic and code reuse.

f. React Router
- Answer: React Router is a standard library for routing in React applications. It enables navigation among different views or components, supports dynamic routing,

and allows the creation of single-page applications with multiple routes without reloading the page.

g. State Management

- Answer: State management refers to the approach of managing and centralizing the state of an application. While React's built-in state is sufficient for many cases, larger applications often use external libraries such as Redux, MobX, or even the Context API to manage global state in a more predictable and scalable way.

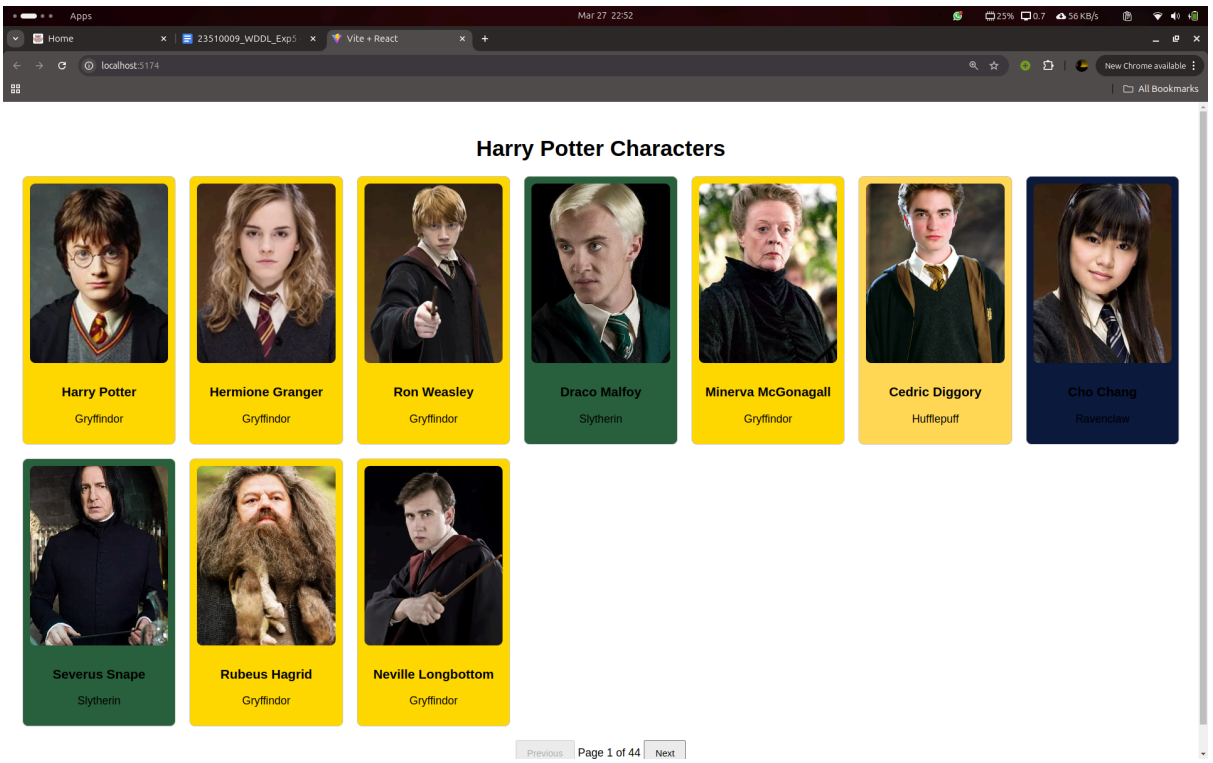
h. React Context API

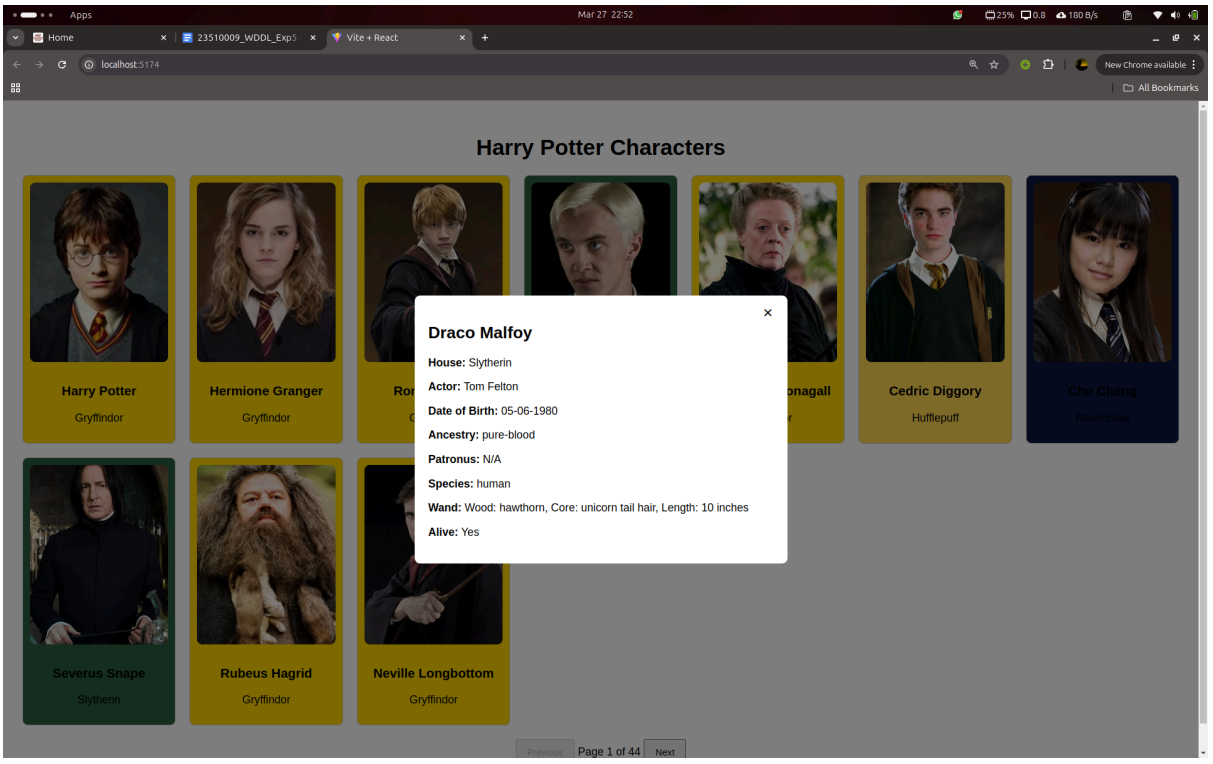
- Answer: The React Context API provides a way to share data (like themes, user information, etc.) throughout the component tree without having to pass props manually at every level. It creates a global context that can be accessed by any component, simplifying state management for certain use cases.

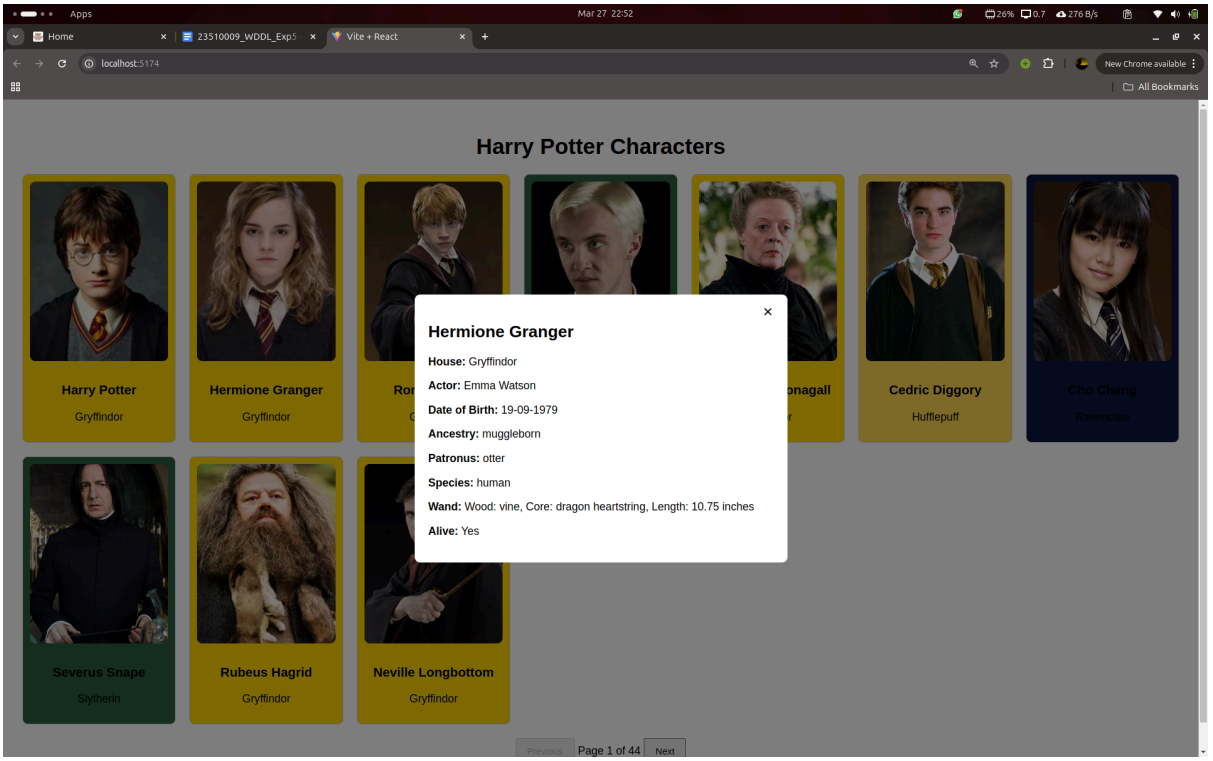
8. How can you optimize the performance of a React application?

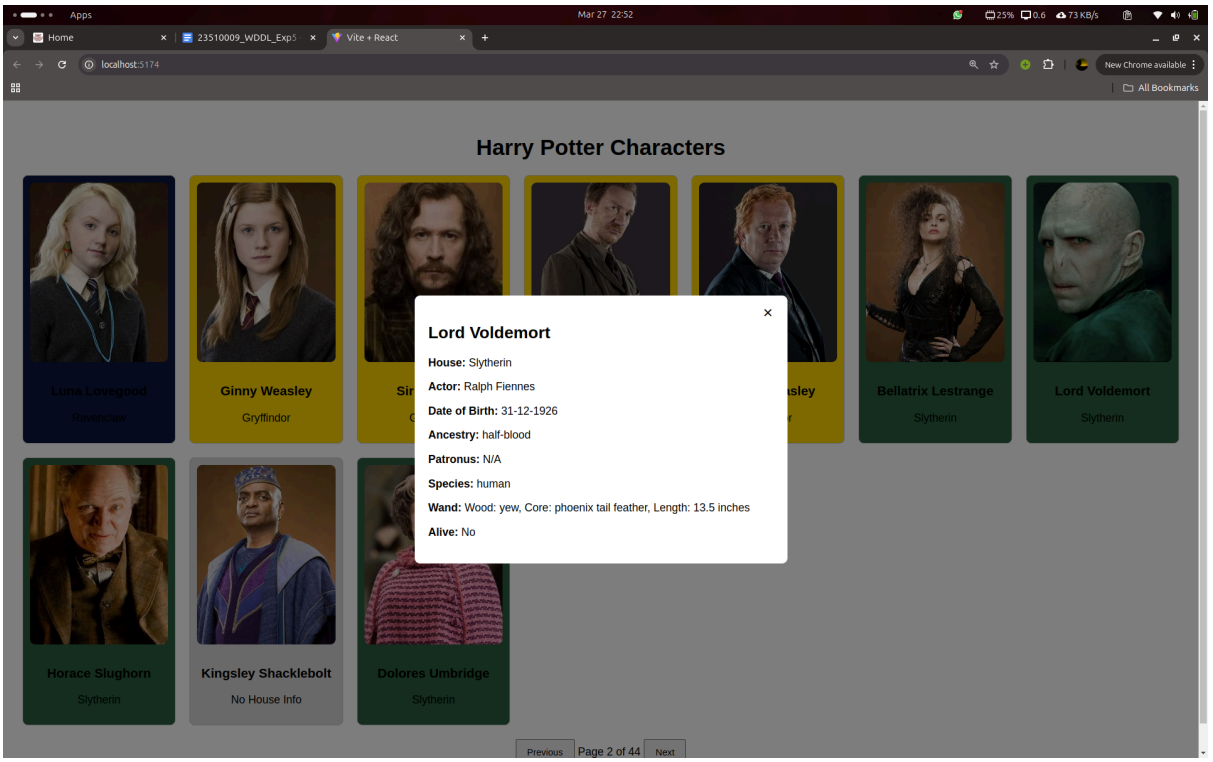
- Answer: Performance optimization in React can be achieved through various strategies:
  - Using React's PureComponent or React.memo to prevent unnecessary re-renders.
  - Implementing code-splitting and lazy loading to reduce initial load time.
  - Using the useCallback and useMemo hooks to memoize functions and values.
  - Leveraging virtualization techniques for rendering large lists.
  - Optimizing asset loading and reducing bundle size.
  - Employing efficient state management practices to minimize component updates.

Problem Statement 1:


Apps
Mar 27 22:52
Home
23510009_WDDL_Exp5
Vite + React
localhost:5174
New Chrome available
All Bookmarks
Harry Potter Characters
Harry Potter
Gryffindor
Hermione Granger
Gryffindor
Ron Weasley
Gryffindor
Draco Malfoy
Slytherin
Minerva McGonagall
Gryffindor
Cedric Diggory
Hufflepuff
Cho Chang
Ravenclaw
Severus Snape
Slytherin
Rubeus Hagrid
Gryffindor
Neville Longbottom
Gryffindor
Previous
Page 1 of 44
Next


Apps
Mar 27 22:52
Home
23510009_WDDL_Exp5
Vite + React
localhost:5174
New Chrome available
All Bookmarks
Harry Potter Characters
Draco Malfoy
House: Slytherin
Actor: Tom Felton
Date of Birth: 05-06-1980
Ancestry: pure-blood
Patronus: N/A
Species: human
Wand: Wood: hawthorn, Core: unicorn tail hair, Length: 10 inches
Alive: Yes
Harry Potter
Gryffindor
Hermione Granger
Gryffindor
Cedric Diggory
Hufflepuff
Cho Chang
Ravenclaw
Severus Snape
Slytherin
Rubeus Hagrid
Gryffindor
Neville Longbottom
Gryffindor
Previous
Page 1 of 44
Next


Apps
Mar 27 22:52
Home
23510009_WDDL_Exp5
Vite + React
localhost:5174
New Chrome available
All Bookmarks
Harry Potter Characters
Harry Potter
Gryffindor
Hermione Granger
Gryffindor
Cedric Diggory
Hufflepuff
Cho Chang
Ravenclaw
Severus Snape
Slytherin
Rubeus Hagrid
Gryffindor
Neville Longbottom
Gryffindor
Hermione Granger
House: Gryffindor
Actor: Emma Watson
Date of Birth: 19-09-1979
Ancestry: muggleborn
Patronus: otter
Species: human
Wand: Wood: vine, Core: dragon heartstring, Length: 10.75 inches
Alive: Yes
Previous
Page 1 of 44
Next


Apps
Mar 27 22:52
Home
23510009_WDDL_Exp5
Vite + React
localhost:5174
New Chrome available
All Bookmarks
Harry Potter Characters
Luna Lovegood
Ravenclaw
Ginny Weasley
Gryffindor
Bellatrix Lestrange
Slytherin
Lord Voldemort
Slytherin
Horace Slughorn
Slytherin
Kingsley Shacklebolt
No House Info
Dolores Umbridge
Slytherin
Lord Voldemort
House: Slytherin
Actor: Ralph Fiennes
Date of Birth: 31-12-1926
Ancestry: half-blood
Patronus: N/A
Species: human
Wand: Wood: yew, Core: phoenix tail feather, Length: 13.5 inches
Alive: No
Previous
Page 2 of 44
Next