Name :- Dhiraj Rajput
Roll no:- 07
PRN :- 1032230441

TY Btech CSF
FSD Lab

classmate
Date
Page

Lab Assignment 01

* **Aim :-** Develop responsive web design using HTML 5, containing a form. Style the pages using CSS, Use of tag selector, class selector and id selectors. Use Inline, Internal and External CSS, Apply Bootstrap CSS.

* **Objective:-**

i) To understand HTML2 tags
ii) To learn the styling of web pages using CSS
iii) To learn Bootstrap Front End Framework.

* **Theory :-**

1) Define Responsive Web Design (RWD). What is its primary goal?

Ans Responsive Web Design (RWD) refers to designing websites to adapt to a user's device. The goal is for a website to retain its optimal usability and appearance regardless of the device it's displayed on. Responsive Web Design (RWD) responds to user needs by adapting to different screen sizes, orientations, layouts and platform. This is accomplished with the use of flexible grids and layouts, responsive images, and CSS media queries.

2) Explain the role of the <meta name: "viewport" ....> tag. Why is this tag essential to RWD?

Ans A Browser's viewport is the area of the web page in which the content is visible to the user. The viewport doesn't have the same size, it varies with the variation in screen size of the device on which the website is visible. For a laptop, the viewport has a larger size

as compared to a smartphone or tablet.

② When a page is not made responsive for smaller viewports it looks bad or even breaks on a smaller screen. To fix this problem introduce a responsive tag to control the viewport. This tag was first introduced by Apple Inc. for safari OS.

③ Syntax:-

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

① The 'width=device-width' in meta tag sets the width of the page to follow the screen width of the device, which will vary depending upon the device.

② The 'initial scale=1.0" in meta tag sets the initial zoom level when the page is first loaded by the browser.

3) How does Bootstrap assist in creating a responsive layout? Describe the concept of a grid system and how it adapts to ~~describe~~ different screen sizes.

Ans Bootstrap is a popular front-end framework that simplifies the creation of responsive website. It provides a collection of pre-written HTML, CSS and Javascript components, with its most powerful feature being responsive grid system.

② Bootstrap grid system divides the horizontal space of page into 12-column grid where we can place content inside these columns, we don't create grid ourselves, but use Bootstraps predefined CSS classes to tell our content

how many of the 12 columns it should span.

③ Grid system is responsive because it has different classes for different screen size "breakpoints", we can assign multiple classes to an element to define its behaviour on different screens.

④ For eg :- on large screen, this dive will take up 4 columns (one-third of screen space. On medium (md) screen, it will take up 6 columns (half of screen spaces). On small (sm) screen takes up 12 columns causing columns to stack vertically

⑤ By these simple classes, developers quickly build complex layouts that automatically reflows and adapt, ensuring content is well-organized and readable

4) Differentiate between Tag, class and ID selectors

Ans

Feature	Tag Selector	Class Selector	ID selector
⊙ Syntax	elementname	.className	#idname
⊙ Scope	Selects all HTML elements of a certain type (eg all H2 tags)	selects all elements that share the same class attribute	Selects only one unique element that has a specific id attribute
⊙ Reusability	Affects all matching tags on a page	Can be used on many different elements on page	Must be unique & can only be used once per page

Feature	Tag Selector	Class Selector	ID Selector
⊙ Specificity	Low: easily overridden	Medium: overides tag	High: Overrides both selectors
⊙ Primary Use	Applying general base styles to all instances of an element	Styling a group of similar elements that should look same	Targetting a single specific structural element on a page - like header or footer

05) Describe the three main ways to apply CSS to an HTML document

Ans Inline CSS :- Inline CSS involves applying styles directly to individual HTML elements using the style attribute. This method allows for specific styling of elements within the HTML document, overriding any external or internal styles

Example :-

```
<p style = " color : white ; font-size: 50px " > Inline CSS </p>
```

i) For quick fixes and small changes that don't require a separate CSS file.

ii) When you need to override other styles for a particular element

iii) If you're working on emails or HTML-based applications where external CSS is not supported.

② Internal or Embedded CSS :- Internal or Embedded CSS is defined within the HTML document's <style> element It applies styles to specific elements. The CSS rule set should be within the HTML file in the head section, i.e. the CSS is embedded within the <style> tag

inside the head section of the HTML file

Example :-

```
<!Doctype HTML>
<html>
<head>
  <style>
   .main { text-align : center ; }
   .nav { color: white; font-size: 90 px }
  </style>
</head>
<body>
  <div class = "main">
    <div class = "nav"> Internal CSS </div>
  </div>
</body>
</html>
```

i) When designing a single-page website

ii) If you need better control over styling than inline CSS.

iii) For small to medium-sized projects where external css might be unnecessary.

③ External CSS :- External CSS contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, ..., etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages

Example :-

HTML

```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="main">
    <div class="yolo"> External css </div>
    <div id="holo"> This shows implementation of
      External css
    </div>
  </div>
</body>
</html>
```

CSS

```
body { background-color: powderblue; }

.main { text-align: center; }

.yolo { color: white; font-size: 50 px; font-weight: bold; }

#holo { font-style: bold; font-size: 20px; }
```

Ground.

i) For large-scale projects where multiple pages share a common design.

ii) When maintainability and scalability are priorities.

iii) To improve website performance and load times using CSS caching.

* Problem Statement :- 1) E-commerce Product Page (Roll number 1 to 12)

* Conclusion :- Thus, learned about various HTML tags their scope, syntax and usuage, came to know about bootstrap, various methods of CSS styling and hence implemented a responsive web design using the concepts used.

30

Name:- Dhiraj Rajput TY Btech CSF
Roll no:- 07
PRN :- 1032230441 FSD Lab

classmate
Date
Page

Lab Assignment-2

* **Aim:-** Develop a web application using javascript to implement sessions, cookies, DOM. Perform validations such as checking for emptiness, only numbers for phone number, special character requirement for password, regular expressions for certain format of the fields etc. Use the mySQL database

* **Objective:-**

i) To understand what form validation is
ii) To learn basic functioning of DOM objects.
iii) To learn how to apply techniques to implement it.

* **Theory:-**

1) Explain the role of regular expression. Why are they a suitable tool for validating data formats like a phone number or checking for the presence of specific characters in a password?

Ans A regular expression is a sequence of characters that define a search pattern, mainly for use in pattern-matching with strings, or string matching, i.e. "find and replace" like operations.

② Regular expressions are a generalized way to match patterns with sequences of characters. It is used in every programming languages like C++, Java and Python, etc.

(e) Phone number validation - Phone numbers have specific formats that can vary across countries, but they generally have certain characteristics that can be validated. Regex can be used to -

i) Ensure the correct number of digits.
ii) Check for the correct use of symbols, like hyphens, parantheses or space.
iii) Validate the presence of county codes or area codes.

For example, ~~a sinup~~

^\+91 [7-9] {1} [0-9] {9} $

This ensures the phone number is in format +917974284[illegible]

(w) Password validation typically require a combination of specific character type (e.g - lowercase, uppercase, numbers, special character) to ensure security. A regular expression can quickly check if the password meets these requirements.

For example, a password regex might check for
i) At least one lowercase letter.
ii) At least one upper case letter.
iii) At least one number.
iv) At least one special character (like @, #, $, etc.)
v) A minimum length of 8 characters

~~For~~ regex for such password might look like

^(?=.*[a-z])(?=.* [A-Z])(?=.*\d)(?=.* [@#$!%*?&]) [A-Za-z\d@$!%*?&#] {8,} $

2) Explain the fundamental difference between a session and a cookie in the context of web application development. How do they work together to maintain a user's logged-in state?

Ans Cookies are small data stored on the client side (browser) as key-value pairs. They are commonly used for session management, user preferences, and behaviour tracking. When a user loads a website, the browser sends the stored cookie with the request, allowing the server to track and personalize the user's experience.

(2) Sessions in Express enable the server to maintain user specific data across multiple requests by storing information server-side and associating it with a unique session identifier. This approach allows for persistent user interactions and state management within web application

(3) Sessions and cookies work together to maintain logged-in state

(i) When a user logs into the application, the server creates a unique session ID for the user, which is stored on the server (often in a session store).

(ii) The server sends this session ID to the user's browser via a cookie. The cookie is stored on the client's side and is sent back to the server with each subsequent request. This allows the server to identify the user during future request without requiring them to log in again.

(iii) Every time the user makes another request to the server, the browser automatically sends the session cookie along with the request. The server can then use the session ID from the cookie to retrieve the user's session data from the server-side storage and verify their identity.

(iv) Expiration - Sessions typically have an expiration time after which the server will no longer recognize the session ID as valid. If the session expires, the user must log in again.

(v) When user logs out, the server invalidates the session ID, and the session data is discarded. The session cookie may also be deleted from the user's browser.

Q3) What is the purpose of performing both client-side and server-side validation? Describe a scenario where relying solely on client-side validation could lead to a security vulnerability.

Ans Client-side validation improves user experience by providing immediate feedback. For example when a user submits a form with missing or incorrect data, the form can highlight the issue without needing to reload the page or send the request to the server.

(2) It can be used as first line of defense, ensuring that some basic validation rules are met

classmate
Date
Page

(3) Server-side validation is essential for ensuring that the data submitted to the server is valid, accurate, and secure. Since client-side validation can be bypassed (user can disable Javascript or manipulate data sent to the server), the server must validate all incoming data before processing it.

(4) Server-side validation ensures that harmful data (eg. SQL injection, cross-site scripting) does not get processed. It acts as the first line of defence against threats.

(5) Imagine a Website that allows users to register by entering a username, email and password. Lets assume the website uses only client-side validation for checking the format of the email & password strength.

(a) A malicious user can disable Javascript in their browser or use developer tools to tamper with the form data before submitting it. They might bypass the email format checker or weak password rules, submitting potentially dangerous or incorrect data.

(b) Even if the user inputs valid data on the client-side, there is still a risks that a hacker might send malicious data.

(c) The server might process this malicious input and execute harmful operations like modifying databases, stealing sensitive information, or triggering malicious code execution.

Q4) Provide a simple example how a Javascript script can interact with the DOM to dynamically change the content of a web page after a user action, such as a form submission.

Ans:

```
<!DOCTYPE html>
<html>
<head>
 <title> DOM Interaction </title>
</head>
<body>
  <h2> Enter Your Name </h2>

  <form id= "nameform" >
   < input type = "text" id= "username" placeholder=
 "Type your name" >
   <button type= "submit"> Submit </button>
  </form>

  <p id ="message"></p>

<Script>
  const form= document. getElementById ('nameform
  const input= document. getElementById ('username
  const message = document - getElementById ('message

  form. addEventListener ('submit', function (event) {
       event. preventDefault ();
       const name= input. value;
       message. textContent =" Hello" + name + "! Welc
 to the site"; });
</script>
</body>
</html>
```

(i) User types his name into the input box.
(ii) When they click submit, JavaScript intercepts the form submission (preventing page reload).
(iii) The script updates the <p>

5) Give the steps for connectivity from Frontend using HTML, CSS, JS to mysql

Ans Frontend - Create a form to collect user using HTML/CSS/JS.

(2) Send data to backend :- Use JavaScripts fetch API or AJAX to send input data via HTTP POST / GET to a backend API.

(3) Backend Server :- A server (Node.js., PHP, Python, etc.) recieves the request.

(4) Database Connection :- The backend uses a MySQL client Library (eg. mysql package in Node.js) to connect to the MySQL database

(5) Query exeution :- Backend executes SQL queries to update, or insert or fetch data.

(6) Send Response :- Backend sends a success / failure response to the frontend.

(7) Frontend upates UI :- The frontend reacts to the response and updates the UI accordingly.

* FAQs :-

i) Write 3 reasons why Form validations are important.

Ans Form validation is the process of making sure that the data submitted through web forms is accurate, consistent, and conforms to specific rules and requirements that are predefined.

(2) If these requirements are not met, or the data is not correctly filled out, the form data will not be accepted or stored on the server.

(3) Form validations stops user from filing wrong data or prevents malicious entities from filing in harmful code or viruses via forms, it acts as the first line of defense for business against erroneous data entry, malicious user, and spam.

(4) Form validation covers the following.

(i) Data Accuracy :- Form validations helps to ensure that the data collected from users is accurate and complete. This is important for business that rely on this data to make decisions.

(ii) Security :- form validation helps to prevent malicious users from submitting harmful data or code. This can help to protect your website and users from attacks.

(iii) User Experience :- It can improve user experience by preventing users from submitting forms with errors or telling them exactly what needs to be fixed. This can save users time and frustrations.

ii) Give an example of how to modify an attribute value using DOM.

Ans

```
<!DOCTYPE html>
<head>
  <title> Modify DOM Attribute </title>
</head>
<body>
  <img id = "myImage" src = "old-image.png"
   alt = "Old Image" width = "200">

  <button onclick = "changeImage()"> Change Image </button>

  <script>
   function changeImage () {
     let img = document.getElementById ("myImage");
     img.setAttribute ("src", "new-image.png");
     img.setAttribute ("alt", "newImage");
   }
  </script>
 </body>
</html>
```

① Initial HTML setup

ⓐ There is an <img> tag with id="myImage", src="old-image.png", alt="Old Image", width="200"

ⓑ A <button> is also present with onclick="changeImage()". It means button is clicked, it will run the changeImage() function.

② Accessing the Element; let img = document.getElement("myImage"). The function grab the <img> element by its id and stores it in a variable called img.

③ Modify the src attribute; img.setAttribute("src", "newimage.png"). This replace the image source. So instead of showing old-image.png, the <img> will now load and display new-image.png.

④ Modifying alt attribute; img.setAttribute("alt", "New Image"). The alternative text is changed from "Old Image" to "New Image". If the image doesn't load, browsers will show "New Image" instead.

⑤ Before clicking the button → You see the image old-image.png. After clicking the button → The image switches to new-image.png and the alt text updates.

iii) What are the different features of JavaScript?

Ans Lightweight and Interpreted- JS is interpreted by the browser directly (no need for compilation). Makes it fast and easy to run a web pages

② Object-Oriented :- Everything in JS is an object (with properties

and methods). It supports objects, inheritance, and prototype (prototype-based OOP instead of classical OOP).

③ Dynamic Typing:- Variables don't require explicit type definition
Ex:- let x = 10; // number x = "Hello"; // now a string

④ First-class functions - functions in JS are treated like variables. They can be passed as arguments, returned from other functions and assigned to variables.

⑤ Event-Driven:- JS responds to user interactions (clicks, mouse movements, keyboard input). Ex :- onclick, onmouseover.

⑥ Asynchronous & single-Threaded:- JS uses a single-threaded event loops. Supports asynchronous operations via callbacks, promises, and async/wait. Ex:- setTimeout() ⇒ console.log ("Hello after 2 seconds"), 2000);

⑦ Client-side Scripting :- Originally designed to run in browsers, making webpages dynamic and interactive. Ex:- form validation, animations, DOM manipulations.

⑧ Sever-side Scripting :- With Node.js, JS can also run on the server. This makes it possible to use one language for both frontend and backend.

⑨ Cross-Browser Compatibility :- Works on all modern browsers (with some differentes handled using standards like ECMAScript)

⑩ Platform Independent - JS code runs in any browser or environment that has a JS engine (like Chrome's V8, Firefox's SpiderMonkey).

(11) Prototype-Based inheritance - Instead of classical class-based inheritance, JS uses prototypes to share properties and methods between objects

(12) Rich libraries & Frameworks - Huge ecosystem: React, Angular, Vue.js (frontend), Node.js, Express.js (backend). Libraries like jQuery, D3.js, etc. simplify tasks.

(13) DOM Manipulation :- JS can access and modify the Document Object Model (DOM). Ex:- document.getElementById("demo").innerHTML = "Hello, World");

(14) Case-sensitive :- JS distinguishes between uppercase and lowercase. Ex myVar and MyVar are two different variables

(15) Supports JSON:- JS Object Notation (JSON) is a lightweight data format used widely in API's. Native support makes data exchange easy.

9/9/25

Name :- Dhiraj Rajput TY Btech CSF

Roll no:- 07

PRN :- 10322304441 FSD Lab

Lab Assignment - 03

* Aim :- Design an interactive front end application using React by implementing templating using components, States, and Props, Class, Events. It must be responsive to scale across different platforms.

* Objective :-

⊙ To develop a responsive, interactive front end application using React.js that effectively demonstrates the fundamental concepts of component-based architecture, state management, and event handling. The application will serve as a practical exercise in building a scalable user interface by implementing templating with components, managing dynamic data with states and props, and handling user interactions with events, ensuring a seamless user experience across various devices and screen sizes.

* Theory

1) Explain the Role of States and Props in React. How do they differ, and what is the primary purpose of each in managing data flow within a component-based application.

Ans State in a built-in object in React components that holds data or information about the component. It is mutable, which means it can be updated within the component using the setState method in class components or the useState hook in functional components

(i) State is local to the component and can't be accessed by child components unless passed down as props.

(ii) It is mutable, meaing it can change over time based on user interactions or API responses.

(iii) When State updates, the component re-renders to reflect the changes.

(iv) Managed using useState in functional components or this.setState in class components

(2) Props:- are used to pass data from a parent component to child component. Unlike state, props are immutable, meaning they can't be modified within the receving component.

(i) Props allow components to be resuable and dynamic.
(ii) Props are read-only and can't be changed by the child component.
(iii) They help in data communication between components.
(iv) Passed as attributes in JSX elements.

Aspect	State	Props
(1) Modification	Can be changed by the component itself	cannot be changed by the receiving component, props are read-only
(2) Communication	Facilitates communication within a component	Facilitates communication between components (from parent to child)
(3) Component Type	State is used in class components (via this.state and this.setState) or functional components (via useState)	Props are used in both class and functional components

Aspect	State	Props
(4) Re-rendering	Changes in state trigger a re-render of the component where the state is modified.	Changes in props cause a re-render of the child component that recieves them, but the parent manages them.
(5) Effect on Parent	Changing state only affects the component where the state is defined	Changing props doesn't affect the parent component directly, the parent controls the props passed to the child.
(6) Responsibility	Managed by the component itself	Managed by the parent component.

(3) Use State when you need to manage data that can change over time within a component (eg from inputs, counters, UI targets)

(4) Use Props when you need to pass data from a parent component to a child component to make components reusable

2) What is a React component? Difference between a Class component and functional component, and discuss the advantages of using a functional component with hooks like useState and useEffect over a class component.

Ans In React, components are reusable, independent code blocks (A functions or a class) that define the structure and behavior of the UI. They accept inputs (props) and return elements that describing what should appear on the screen.

(2) Key Concepts of React Components.
- Each component handles its own logic and UI rendering.
- Components can be reused throughout the app for consistency.
- Components ~~can be reused~~ accepts inputs via props and manage dynamic data using state.
- Only the changed component re-renders, not the entire page.

(3) Functional Components :- They are simpler and preferred for most use cases. They are JS functions that return React elements. With the introduction of React Hooks, functional components can also manage state and lifecycle events.

- Stateless or Stateful: Can manage state using React Hooks.
- Simpler Syntax : Ideal for small and reusable components.
- Performance : Generally faster since they don't require a 'this' keyword.

(4) Class components :- They extend React.Component. They include additional features like state management and lifecycle methods.

- State Management - State is managed using this.state property.
- Lifecycle Methods - Includes methods like componentDidMount, componentDidUpdate, etc.

Feature	Functional Component	Class Component
① State Management	It can use Hooks like useState, useReducer	It uses this.state & this.setState()
② Lifecycle Methods	It uses useEffect Hook for lifecycle methods	It uses traditional lifecycle methods like componentDidMount, componentDidUnMount
③ Rendering	Return JSX directly inside function	Use a render() method to return JSX
④ Performance	Faster and more lightweight due to simpler structure	Slightly heavier due to the overhead of class instances
⑤ Hooks	Can use React hooks (useState, useEffect etc.).	Can't use hooks, relies on lifecycle methods and state
⑥ This Keyword	Doesn't use this keyword	Use this to access props & state
⑦ Code Complexity	Less boilerplate code, easier to write & understand	More boilerplate code, especially for state & methods
⑧ Event Handling	Simple and direct event handling	Requires methods binding for event handling.

3) Describe the concept of "templating using components" in React. Why is this approach considered superior to traditional web development methods that rely on monolithic HTML files?

Ans In React, "Templating using components" help us to build user interface (UIs) by composing many small components (Button, Card, Header etc). Each component encapsulates markup (template), style and behaviour.

(2) The app so formed is then a tree of components. Its superiority to monolithic HTML files and described in point

(i) Reusability - components can be reused across pages (project)
(ii) Encapsulation - Each component manages its internal state styles reducing global side effects
(iii) Maintainability - Small focused components are easier to understand, test and refactor.
(iv) Declarative update - React describes what UI should look like for a given state. React handles DOM changes efficiently (less manual DOM manipulation)
(v) Composition - Complex UI's are built by composing simple pieces, following the same mental model as functions.
(vi) Performance - Virtual DOM differs update only what's necessary.
(vii) Developer engonomics - Hot reloading, clear props API, and hooks make evolving UI factor than editing huge HTML

4) How do you handle events in React (e.g. a button click)? Provide a simple code snippet to demonstrate how an event handler is defined in a component and how it can be used to update the component's state.

Ans Event handlers are functions passed to JSX event props, they can even update state; code snippet is given below:-

```
import React { useState} from "react";
function Likebutton (){
    const [liked, setLiked ] = useState (false);
function handleClick () {
        setLiked (props => ! prev);
    }
return {
    < button onclick={handleClick }> { Like : "Unlike"
:like' } </button>
};
}
export default Likebutton;
```

5) What is a responsive web design, and why is it crucial for modern applications? Describe how you would implement a responsive design in a React application using CSS media queries or a CSS-in JS library.

Ans A Responsive Web design (RWD) ensures a website adopts to different screen sizes and devices like desktop, tablet & mobile, etc. It improves user experience across all platform

- Essential for accessibility and search engine optimization
- Reduces the need for multiple device-specific versions

implementing Responsiveness in React.

CSS media Queries

```
.container {
    padding : 20px;
}
@media (max-width: 760px) {
    .container {
        padding : 10px
    }
}
```

CSS in JS

```
import styled from 'style-components';

const Container = styled.div
  padding: 20 px;

  @media (max-width : 760px) {
    padding : 10px;
  }
}

export default function App () {
    return <Container> Responsive Content </Container>;
}
```

Conclusion :- Thus learned and implemented some of the big principle of React development. Components make the UI modular and reusable, state & props execute and enable efficient data flow and dynamic updates, Event handling makes applications interactions ; while RWD helps to ensure accessibility across platforms on devices

Name :- Dhiraj Rajput
Roll no :- 07
PRN :- 10322304411

TY Btech CSF
FSD Lab

Lab Assignment - 4

* Aim :- Enhance ~~web~~ web page developed in earlier assignment by rendering Lists and Portals, Error Handling, Routers and Style with React CSS also make it a responsive design to scale well across PC, tablet and Mobile Phone

* Objective :-
- Enhance User Interface and Experience
- Improve Application Robustness and Navigation.

* Theory

1) How do Lists and keys work in React?

Ans In Lists, React makes it easier to render multiple elements dynamically from arrays or objects, ensuring efficient and reusable code.

2) Since nearly 85% React project involve displaying data collection - like user profiles, product catalogs, or tasks - understanding how to work with lists

3) To render a list in React, we will use the JavaScript array map() function. We will iterate the array using map() & return the required element in the form of JSX to render the repetitive elements.

4) For example,

```
function App() {
    const items = ['Apple', 'Banana', 'Cherry'];
```

```
return (
<div>
    <h1> My fruit List </h1>
    <ul>
      { items.map ( ( item, index) => (
          <li key = {index} > {item} </li>
      ))}
    </ul>
  </div>
  );
}
export default App;
```

① The .map() function loops over the items array.
② Each item is rendering inside an <li> tag.
③ The key prop should be added to uniquely identify each list item.

④ A key serves as a unique identifier in React, helping to track which items in a list have changed, been updated, or removed. It is particularly useful when dynamically creating components or when users modify the list

⑥ When rendering a list, you need to assign a unique key prop to each element in the list. This helps React identify which elements have changed, been added, or been removed

④ Example,

```
const numbers=[1, 2,3,4,5];
const updatedNums = numbers.map ((number, index)=>
  <li key = {index}> {number} </li> );
```

- map() loops over the numbers array
- For each item, it creates an <li> element with the value of the number
- The key for each <li> is assigned as the index of the item in the array.

2) What is a React Portal and when would you use one?

Ans Portals in React come up with a way to render children components into a DOM node which typically occurs outside the DOM hierarchy of the parent component.

(2) Before React Portals, it was very difficult to render the child component outside the hierarchy of its parent component. Every single React component in the React application falls under the Root element

(3) But the React portal concepts provides us the ability to break out of this dom tree and render a component onto a dom node that is not under this root element.

(4) Doing so breaks the convention where a component needs to be rendered as a new element and follows a parent-child hierarchy.

(5) Portals are commonly used in modal dialog boxes, hover cards, loaders, and popup messages. Below is the syntax of React Portals.

ReactDOM.createPortal(child, container).

(6) We mainly need portals when a React parent component has a hidden value of overflow property (overflow: hidden) or z-index style, and we need a child component to openly

come out of the current tree hierarchy.
Following are the examples when we need the reactportals:
① Dialogs
② Modals
③ ToolTips
④ Hovercards
In all these cases, we're rendering elements outside of the parent components in the DOM tree.

③ Discuss the importance of Error Boundaries in React?
Ans Error Boundaries are a powerful tool in React for handling errors gracefully and preventing them from crashing the entire application. Imagine an error boundary like a safety net surrounding specific components

② When an error occurs within the wrapped components, the error boundary catches it and prevents it from propagating further. Instead of crashing, the error boundary displays a custom fallback UI, often informing the user and providing potential solutions.

③ Use error boundaries when you want to prevent crashes in your app caused by unexpected errors in certain parts of your UI. It helps keep your app running smoothly even if something goes wrong.

④ Example :- Imagine you have a component that fetches data from a server. If the server is down or there's an issue with the data, instead of crashing the whole app, you can use an error boundary to handle that error gracefully and show a user-friendly message instead.

4) How does React Router enable Single Page Application (SPA) functionality?

Ans SPA (Single Page Application), loads a single HTML page, dynamically updates content without full page reload; thus improves speed and user experience like a desktop application.

(2) Browser navigation events and updates the URL and renders the matching component without reloading page. Uses <BrowserRouter>, <Routes>, <Route> and <Link> for declarative routing.

(3) Example- import {BrowseRoute, Routes, link} from "react-route"

```
function App() {
  return ( <BrowserRouter>
<nav>
 <link to:"1"> Home </Link>
 <link to="/about"> About </Link>
  </nav>
  <Router>
  <Route path="/" element= {<h1> Home </h1>}>
  < Route path :="/about" element= {<h1> About
</h1>}>
  </Route>
  </BrowseRouter>
}
```

Q5) Explain the different ways to style a React Application.

Ans Styling components is crucial for creating visually appealing applications.

② Types of styling a React Application are :-

i) Plain CSS - Import css file or use CSS modules for scoped classnames

```
import "./app.css";
import styles from "./App.module.css";
<div className={styles.button}>Click</div>
```

ii) Inline style :- Use JS objects for styles

```
<div style={{ color: "blue", padding: "10px" }}>
Hello </div>
```

iii) CSS in JS :- Write CSS directly inside JS. Scoped to Components

```
import styled from "style-components";
Const button = styled.button`
  background: blue;
  color: white;
  &:hover { background: darkblue; }`;
```

iv) Utility-first CSS :- Use prebuilt utility classes

```
<button className="bg-blue-500 text-white">Click
</button>
```

26/9

Conclusion :- Thus implemented & understood the concept of lists, keys, Portals, error boundaries, React-Router & styling techniques in React. There enhancements collectively improve both user experience & application robustness.

Name:- Dhiraj Rajput
Roll no:- 07
PRN:- 10322304441

TY Btech CSF
FSD Lab

Lab Assignment - 5

* Aim:- Develop a Responsive Web design using Express Framework to perform CRUD operations and deploy with NodeJs .use MongoDB.

* Objective:-
1. Develop a Full-stack Web Application
2. Demonstrate Backend Development and Deployment Proficiency.

* Theory:-

1) What is the role of Express.js as a web framework for Node.js?

Ans Express.js is a minimal and flexible Node.js web application framework that provides a list of features for building web and mobile applications easily.

2) It simplifies the development of server-side applications by offering an easy-to-use API for routing, middleware and HTTP utilities.

3) Built on Node.js for fast and scalable server-side development.
4) Simplifies routing and middleware handling for web applications
5) Supports building REST APIs, real-time applications, and single-page applications
6) Provides a lightweight structure for flexible and efficient server-side development.

③ For example;

```
const express = require ('express');
const app = express ();

app.get ('/', (req, res) => {
    res.send ('Welcome to Express');
});

app.listen (3000, () => {
    console.log ('Server is running');
});
```

2) Explain the concept of CRUD operations in the context of a web application.

Ans CRUD stands for Create, Read, update and Delete, the four basic functions for interacting with data in any web application

	Operation	HTTP Method	Description	Example in a Web app
①	Create	POST	Adds new data to db	users sign up → data saved in DB
②	Read	GET	Retrieve Data from db	View list of products fetches from DB
③	Update	PUT / PATCH	Modify existing data in db	edit users profile update Name
④	Delete	DELETE	Remove data from db	Delete product from catalog → remove Name

For eg:-

```
app.post ('/users', async (req, res) => {
    const user = new user (req, body);
     await user.sav ();
     res.send (user);
    });

app.get ('/users', async (req, res) => {
     const users = await user.find();
      res.send(users);
   });

app.put ('/users', async (req, res) => {
     const user=await user.findByIdAndUpdate (req.paras.id
     req.body)
       { new: true}
     res.send (user);
   });

app.delete ('/user/:id", async (req, res) => {
      await user.FindByIdAndDelete (req.param.id);
       res.send (users);
      });
```

Q3) Why is MongoDB a suitable choice for this project?

Ans MongoDB is NoSQL, document oriented database that stores data in flexible JSON like documents rather than rigid rows and columns like traditional SQL DB.

(2) Reasons for its suitability are :-

i) Schemaless structure :- No fixed schemas, they easy to modify data structures as the application evolves

ii) Scalable :- Handles large amounts of data efficiently, supports horizontal scaling (via sharding).

iii) JSON compatibility :- Stores data in BSON, perfect for the application.

iv) High performance :- Optimized for read/write operations, making CRUD operations very fast

v) Integration with Node.js :- Libraries like Mongoose simplify communication between Node.js and MongoDB

vi) Cloud support :- Easily Deployable on MongoDB Atlas for production use.

Q4) What steps are involved in deploying a Node.js and Express. application?

Ans Deploying means making our web application available on the internet for users. A common approach is using cloud platform like Render, AWS or DigitalOcean.

Typical steps :-

(1) Prepare the Application :-

(a) Ensure package.json has all dependiences.

(b) Use environment variables for sensitive data.

(c) Add a start script in package.json → "start": "node server.js";

(2) Setup Production database
- Use MongoDB Atlas or any other cloudbase services
- Update your connectors string in .env file

(3) Push code to Github.
- Create a repository → commit and push your code.

(4) Deploy on hosting platform
- for ~~*~~ heyroku →

git push heroku main
heroku config.set MongoDB URL → your db connection string
This app gets a live URL after deployment

(5) Testing & monitoring :-
- Checks logs using heroku logs --tail or equivalent tools
- set up monitoring and errors tracking (with LogRocket, etc).

* Conclusion :- Express.js plays a crucial role in simplifying server side development, while CRUD operation form the foundation of dynamic application. MongoDB is well ~~visited~~ suited due to its flexible, NoSQL, schemaless, BSON design. Thus learned and implemented about those tools that together enable the creation of a robust,

26/9/25