Name: Kaashvi Gupta
PRN: 10322313101
Roll: 29
Batch: TY CSF(AL)

FSD Laboratory - 01

AIM: Develop responsive web design using HTML5, 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:

1. To understand HTML tags
2. To learn the styling of web pages using CSS
3. To learn Bootstrap Frontend framework.

Theory:

1) Define Responsive Web design (RWD). What is primary goal?
RWD is a web design approach that ensures a website's layout and content automatically adjust to fit different screen sizes and devices, such as desktops, tablets and smartphones

Primary Goal: to provide an optimal viewing and interaction experience for users by ensuring readability usability & accessibility without requiring separate designs for different devices.

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

The <meta name = "viewport" content = "width = device width, initial-scale = 1.0"> tag controls how a webpage is displayed on mobile devices.

- Role: It sets the visible area (viewport) of the web page and scales it to match the device's screen width.

- Importance in RWD: Without this tag, mobile browsers might render pages at a default desktop width forcing users to zoom & scroll horizontally. This tag ensures the layout adapts to the device's screen size.

3) How does bootstrap assist in creating a responsive layout? Discuss the concept of a grid system & how it adapts to different screen sizes.

- Bootstrap's Role: Bootstrap is a popular front-end framework that includes pre-built .css and JavaScript components for creating responsive websites quickly.

- Grid system: Bootstrap uses a 12-column grid layout that allows developers to divide the page into rows & columns.

Adaptation:

- On small screens, columns stack vertically
- On medium/large screens, columns can appear side-by-side
- This adaptability ensures consistent design

Name: Kaashi Gupta
PRN: 1032221201
Roll no. 29
Batch: TYCSF (A1)

4. Differentiate between Tag, Class & ID selectors

Selector Type	Syntax Example	Description	Usage
Tag selector	p{ color: Red; }	Select all elements of a given HTML tag.	Use when styling all elements of a certain type
Class Selector	.highlight { background: Yellow; }	Selects elements with a specific class attribute	Use for styling multiple elements with the same class
ID selector	#header { fontsize: 20px; }	Selects a single element with a specific ID attribute	Use for styling one unique element on a page

5. Describe the 3 main ways to apply CSS to an HTML document.

1. Inline CSS:
- Defined directly in the HTML element's style attribute
- Example <p style = "color: blue;"> Text </P>
- Used for quick, one-off changes

2. Internal CSS:

- written inside a <style> tag within the <head> section of the HTML

eg:

```
<style>
p{color: green;}
</style>
```

- used for styling a single HTML document.

3. External CSS:

- stored in a separate .css file and linked using <link rel="stylesheet" href="style.css">

eg.

```
<link rel="stylesheet" href="style.css">
```

- used for consistent styling across multiple pages.

Problem statement - 3. Event Registration form (Roll no. 25 to 36)

Conclusion - Thus learned about various HTML tags their scope, syntax and usage come to know about bootstrap. various web design using the concepts used.

Kaashvi Gupta
24-CSF
10322313101
29

Lab Assignment-2

AIM: Develop a web application using JS to implement cookies, session DOM. Perform validations such as checking for emptiness, only numbers for phone no., special character req. for passwords, regular expressions for certain format of the fields etc. Use mySQL as database.

Objective:

1. To understand what FORM validation is
2. To learn basic functioning of DOM objects
3. To learn how to apply various techniques to implement it

Theory:

Explain the role of regular expressions. Why are they a suitable tool for validating data formats like a phone no. or checking for the presence of specific characters in a password?

Regular expressions (regex) are patterns used to match and manipulate strings. Their major role is that they help in validating input formats and searching specific patterns within Data.

Example Use Cases:

① Phone no. validation: Ensures the input follows a pattern like ^\d{10}$ (10 digits).

② Password check: Ensures presence of at least one uppercase, one lowercase, one digit and one special character.

Regex is suitable because it is compact fast & flexible for checking data formats.

Explain the fundamental diff. bet a session and a cookie in context of web application development. How do they work together to maintain users logged-in state.

Cookie	Session
① Stored on the client side	① Stored on the server side
② Holds small pieces of data like session IDs or user preferences	② Identified using a session ID usually stored in cookie, holds sensitive data like logged-in status

When user login, the server creates a session and sends the session ID to the browser inside a cookie. The browser then sends this cookie back on every request, allowing the server to recognise the logged in user.

Q.3) What is the purpose of performing both client & server side validation? Describe a scenario where relying solely on client side validation could lead to a security vulnerability.

Client side validation - Done in the browser using Java Script. Prevents unnecessary server requests eg. if a field is empty it gets checked

Server-side validation - Done on the server. Ensures security since the client side can be bypassed.

Scenario: if only client-side validation is used, a hacker can disable JS in the browser and send malicious data via tools like Postman.

Q.4) Provide a simple example of how a JavaScript can interact with the DOM to dynamically change the content of a web page after a user action, such as for submission.

In the below example when the button is clicked the paragraph text changes dynamically

```
<html>
```

```
<body>
<p id="message"> Hello user ! </p>
<button onclick="changeText()"> Click Me </button>
<script>  function changeText() {
          document.getElementById("message").innerHTML=
          "Welcome Kaashni" ; }
</script>
</body>
</HTML>
```

5) Give the steps for connectivity from frontend using HTML CSS JS to mysql.

1. Frontend (HTML, CSS, JS) - Take user input (eg. form data)
2. AJAX / Fetch API - Sends data to backend without reloading page
3. Backend (Node.js, Python, etc) - Receive request, validate and connect to MySQL.
4. Database (MySQL) - Perform queries (insert, select, update)
5. Backend Response - Sends results (like success or error) back to Front
6. Frontend update - Dynamically update UI (eg. show msg)

FAQs

① Give 3 reasons why form validation are important.

- Prevents incorrect and incomplete data entry
- Enhances security (eg. blocks SQL injection, XSS)
- improves user experience (real-time error messages).

② Give an example of how to modify an attribute using DOM.

```
<img id="myImg" src="old.png">
<script>
document.getElementById("myImg").setAttribute
("src","new.png");
</script>
```

3. What are the diff. features of JavaScript?

① Lightweight & interpreted (runs directly in browser without compilation)

② Event driven & dynamic (responds to user interactions)

③ Object oriented (supports objects, inheritance, and prototypes)

④ Cross-Platform (runs on all major browsers).

⑤ versatile (used for frontend, backend with Node.js and even mobile apps).

Conclusion - The assignment shows how JavaScript with DOM, sessions and cookies can create interactive web apps. Validations ensure data accuracy and security while MySql integration enables reliable data storage together, they build a secure and user-friendly application.

21/9/25

Kaashi Gupta
29
10322231301

Lab Assignment-03

AIM: Design an interactive front end application using React by implementing using components, States and Props, Class, Events. It must be responsive to scale across diff. platforms

Objectives:

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 pratical 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 state and Props in React. How do they differ, and what is primary purpose of each in managing data flow within a component-based application.

Role of state and Prop in React

① **State**

- It is an internal data storage specific to a component
- It is mutable and can be updated using setState (in class components) or useState (in functional components)

- Changes in state trigger re-rendering of the component
- eg: keeping track of a counter, form inputs or t

② Props

- They are read-only data passed from a parent component to a child component
- They allow components to be reusable and communicate with each other.
- eg. passing a title or color to a button com

Difference:

State → managed inside a component (local, mutable)

Prop → passed from parent to child (immuta

Purpose: Together, state and props enable unidirecti data flow, ensuring predictable and maintain applications.

2. What is a React component? Differentiate bet. a cla component and a functional component, and disc the advantages of using a functional component wi hooks like useState & useEffect over a class compon

- A React component is a reusable, independent piece of UI. It can be functional or class-based.

Class-component:

- Uses ES6 classes
- Has lifecycle methods (componentDidMount, componentDidUpdate, etc)
- Manages state using this.state and this.setState()

```
class Welcome extends React.Component {
  constructor(props) {
    super(props);
    this.state = {message: "Hello"};
  }
  render() {
    return <h1>{this.state.message}, {this.props.name}
    </h1>;
  }
}
```

Functional Component: A plain JavaScript function returning JSX. Uses hooks (useState, useEffect) for state and lifecycle behavior.

```
function Welcome({name}) {
  const [message, setMessage] = React.useState("Hello");
  return <h1>{message}, {name}</h1>;
}
```

Advantages of Functional components with Hooks:
- Simpler and more concise syntax.
- Easier to read and test
- Better performance in many cases.

3. Discribe the concept of "templating using components" in React why this approach considered superior to traditional web dev methods that rely on monolithic HTML files.

- Instead of writing one large HTML file, React splits UI into reusable components (buttons, navbar, etc)

- Each component defines its own structure (JSX) and behavior.
- Components can be nested and reused, making the code modular and scalable

It works well:

① Reusability - a button or card can be reused anywhere

② Maintainability - changes in one component don't break the whole app.

③ Dynamic rendering

4. How do you handle user events in React? Provide a simple code snippet to demonstrate how an event handler is defined in a component & how it can be used to update the component's state.

React uses event handlers similar to HTML but in camelCase and passed as functions.

eg. Button click to Update State.

```
import React, {useState} from "react";
function Counter() {
  const [count, setCount] = useState(0);
  const handleClick = () => {
    setCount(count + 1);
  };
  return (
    <div>
      <p> You clicked {count} times </p>
      <button onClick={handleClick}> Click </button>
    </div>
  );
}
export default Counter;
```

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

Responsive Web Design (RWD) ensures a website or app adjust its layout and elements automatically to fit different devices (desktop, tablet, mobile).

Importance

- improves user experience across devices
- Essential for mobile first development
- Increase accessibility and reach.

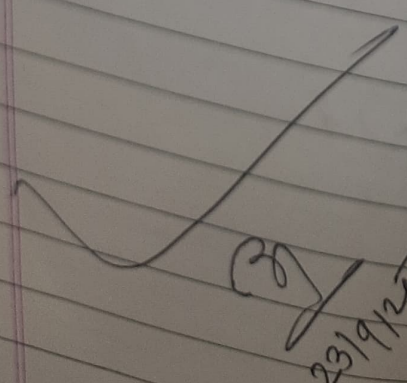
Implementation in React

1) CSS Media Queries:

```
.container {
    padding 20px;
}
@media (max-width: 600px) {
    .container {
        padding: 10px;
        font-size: 14px;
    }
}
```

2. CSS-in-JS (styled-components / emotion)

```
import styled from "styled-components";
const Box = styled.div`
  width: 80%;
  @media (max-width: 768px) {
    width: 100%;
  }
`;
```



Kaashvi Gupta
29
10322313O1

Lab Assignment-04

AIM: Enhance webpage 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

Objectives:

- Enhance User interface and experience
- Improve Application Robustness and Navigation

Theory:

① How do lists and keys work in React?

Lists

- In React, we often display multiple elements using lists (eg. rendering an array of data as <li> items)
- This is usually done with the map() function in JS

Keys

- Keys are special attributes that help React identify which items in a list have changed, been added or removed.
- A unique key (like an ID) must be assigned to each list element

eg.
```
const items = ["Apple", "Banana", "Cherry"];
const list = () => (
  <ul>
    {items.map((item, index) => (
```

<li key={index}>{item}.
))}

);

It important as key improve performance by allowing react to re-render only changed elements instead of the entire list.

2. What is a React Portal and when would you use it?

A React Portal allows you to render children outside into a DOM node outside of the parent component's hierarchy.
It is created using ReactDOM.createPortal(child, ...)

It is used when you need to render a component that visually "escapes" its parent container. eg. Modal dialog boxes, Tooltips, dropdown menus

eg. import ReactDOM from "react-dom";
function Modal ({children}) {
return ReactDOM.createPortal(
{children}</...
document.getElementById("modal-root...
);
}

3. Discuss the importance of Error Boundaries in
- Error Boundaries are special React components that catch JavaScript errors anywhere in their child tree

- They prevent the entire app from crashing by showing a fallback UI
- Implemented using: Class components with componentDidCatch and getDerivedStateFromError.

eg.

```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }
  static getDerivedElement
  static getDerivedStateFromError() {
    return { hasError: true };
  }
  componentDidCatch(error, info)
  console.log(error, info);
}
render() {
  return. this.state.hasError? <h2> Something went
  wrong! </h2>: this.props.children;
  }
}
```

Importance

1. Protects the app from unexpected crashes
2. Provides a way to log and debug errors
3. Enhances user experience by showing fallback messages instead of a blank screen.

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

SPA: In a single Page Application, navigation bet. pages does not reload the entire browser page. In only the required components are updated dyn

React Router: • Provides <Routes>, <Route> and <BrowserRouter> for handling routing.
• Uses the History API to manipulate the URL without refreshing the page.
• Enables navigation between components based the current path
Conclusion: React Router makes React apps behav multi-page apps while actually being one fix improving performance and user exp

5. Explain the diff. ways to style a React Appli

① Inline styling: Add style directly in JSX u objects
eg. <h1 style={{color: "blue", fontSize: "20px"}}>Hello</h1>

② CSS stylesheets: Import external .css files
eg. import "./App.css";

③ CSS modules: Scoped CSS with unique class nam to avoid conflicts
eg. import styles from "./App.module.css";
<h1 className={styles.title}>Hello</h1>

④ Frameworks/libraries: Use Bootstrap, Tailwind Material UI for responsive components.

LAB ASSIGNMENT-05

AIM: Develop a responsive web design using Express framework to perform CRUD operations and deploy with Node Js use MongoDB.

Objective:
- Develop a full Stack web Application
- Demonstrate Backend Development & Deployment proficiency.

Theory:

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

Express.js is a lightweight, flexible and minimal web application framework built on top of Node.js

Role:

① Simplifies serverside ~~deploy~~ development by providing utilities for handling HTTP requests and responses

② Offers built-in middleware and the ability to define custom middleware for processing requests

③ Enables easy creation of RESTful APIs

④ Provides routing features to handle different URL endpoints (Get, Post, Put, Delete).

Conclusion: Express.js acts as the backbone of many Node.js applications by making server side coding faster, easier and more structured.

(2) Explain the concept of CRUD operations in the context of a web application
CRUD fundamental operations for managing data
C → Create: insert new data into database
R → Read: Retrieve existing data
U → Update: Modify existing data
D → Delete: Remove data

Mapping to HTTP methods:
- Post → Create
- Get → Read
- Put/Patch → update
- Delete → delete

Conclusion: CRUD operations are the foundation of dynamic web applications, allowing users to interact with data.

(3) Why is MongoDB a suitable choice for this project?

MongoDB is a NoSQL, document oriented database
- Suitability:
1) Schema-less (flexible) → stores data as JSON like documents, perfect for applications with dynamic or evolving data
2) Scalable → handles large amounts of data and high user loads
3) Fast Development → Easy integration with through libraries like Mongoose
4) Supports CRUD operations efficiently with rich query support.

6. High Availability: Built in replication and sharding

4. Steps involved in Deploying a Node.js and Express Application

1. Prepare Application
   - Ensure package.json is set up with dependencies
   - Add a start script ("start": "node server.js").
   - Use env. variable for sensitive data (like DB credentials).

2. Choose Deployment Platform.
   eg. Vercel, AWS, Render

3. Set Up Server
   ~~npm install~~
   - Install node.js on the hosting machine
   - Upload project files or connect via Git

4. Install Dependencies
   - npm install

5. Configure Database
6. Run and Test application
   - npm start
7. Enable Process Management
8. Configure Reverse Proxy.