Assignment - 01

- Aim: Develop responsive webpage design using HTML5, containg form. Style the web-pages using CSS, Use of tag selector and id selector. Use Inline, Internal and External CSS, Apply Bootstrap CSS.

- Objectives: 1) To understand HTML tags
  2) To learn styling using CSS
  3) To learn Bootstrap front end framework

- Theory

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

→ Responive Web design is an approach to web development that creates a single website capable of adapting its layout & content to fit various screen sizes & devices, such as desktops, laptops and smart phones.

Primary Goal of RWD

The primary of RWD is to provide an optimal viewing & interaction experience for every user regardless of what device they're using.

This means the user shouldn't have to resize, pan or scroll excessively to view a website. By using a flexible grid system, fluid images & ~~CS~~ CSS media queries, RWD ensures that the website design remains accessible, readable & functional across devices, thereby eliminating the ~~need~~ need for seperate mobile version of site.

Q.2 Explain the role of <meta name="viewport" tag, why is this essential for RWD?

→ The meta viewport tag plays a ~~crucial~~ crucial role in responsive web design by giving instruction to browser on how to control the pages dimensions & scaling.

The viewport is the visible area of a webpage on screen. Because device come in various sizes. The viewport tag ensures the web page display correctly on all of them. The tag's key properties are.

- width = device-width: This sets the width of page to match the ~~screen's~~ screen's width. This is fundamental property for RWD as it prevents browser from browser using ~~default~~ default using large viewport that will require user to zoom out to view entire page.

PAGE No.
DATE / /

- initial-scale = 1.0 : This sets the initial zoom level of page when its first loaded. A value of 1.0 means there is no-zoom, ensuring the webpage is displayed at 100% of its intended size & preventing a zoomed-out view on mobile devices.

By including viewport tag, we're saying to browser, "The width of my site is same as of device & at 1:1 scale". This simple instruction enables RWD.

Q.3 How does ~~boot strap~~ bootstrap assist in creating a responsive layout? Discuss the concept of grid system and how it adapts to diff. screen sizes?

-> Bootstraps grid system enables responsive, mobile friendly layout using a 12-column structure.

1. Grid Basics

- Structure: Content is placed in .row & .col elements.
- Classes:
  - .col : Extra small devices, < 576 px
  - .col-sm : Small devices, ≥ 576 px
  - .col-md : Medium devices, ≥ 768 px
  - .col-lg : Large devices, ≥ ~~1200 px~~ 992 px
  - .col-xl : Extra large devices, ≥ 1200 px

2) Responsive Behaviour:

- Fluid Layouts: Columns automatically resize to fill available space
- Break point: We can specify different column width at different screen size.
- Stacking & Allignment: On smaller devices, columns stack vertically, on large devices they can be side-by-side.

3) How it adapts to screen size:

Bootstrap uses CSS media queries to apply different grid configuration depending on device width on the mobile-first principle, styles starts with smaller screen & scale up. If a column has no width set for smaller breakpoint it default to "width: 100%". As screen width increases past a break point, the layout adjusted without needing seprate code.

eg.
```
<div class="container">
    <div class="row">
        <div class="col-12 col-md-8">Main</div>
        <div class="col-12 col-md-4">Side</div>
    </div>
</div>
```

Q.4 Differentiate bet^n class, tag & Id selector.

->

	Tag	Class	ID
Applies to	All element matching the HTML tag	All elements with given class attribute	One element with that unique id.
Uniqueness	Not unique, applies to all tags	Reusable on multiple elements	Must be unique in HTML page
Specificity level	Low	Medium	High
Syntax	tagname	.classname	#id

eg.	h1 {
	font-size: 2px;
	}
		.header: {
		color: blue;
		}
			# comment {
			color: red
			}

Q.5 Descibe 3 main ways to apply CSS on HTML do

-> 1) Inline CSS

- Definition: CSS styles are applied directly inside an HTML ~~tag~~ element using style attribute.
- Scope: Affects only that specific element

- Pros: Quick for small changes or testing
- Cons: Not reusable, clutters HTML, hard to maintain

eg.

```
<p style="color: blue;">Inline</p>
```

2) Internal

- Definition: CSS is placed inside <style> tags in document's <head>
- Scope: Applies only to that page
- Pros: Keeps CSS in one page for that page
- Cons: Not reusable across multiple pages

eg.

```
<head>
  <style>
    p { color: blue; }
  </style>
</head>
```

) External

- Definition:- CSS rules are stored in a seprate .css file & linked with <link> in the HTML <head>

PAGE No.
DATE / /

- Scope: Can style multiple HTML pages
- Pros: Clean HTML, reusable styles, easier maintenance
- Cons: Requires an extra file request from the server.

eg.

```
<head>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
p { color: blue }
```

- Conclusion: In this assignment, I learned how to build responsive web pages using HTML, CSS & bootstrap. I understood the use of different CSS selectors, ways of applying CSS & how viewport tag within Bootstrap grid system ensures layout to give size.

22/8/25

Assignment -02

- Aim : Develop a web application to implement Session, cookies, DOM, Perform validation like checking for emptiness, only no. for phone, Special char. in password, regular expression for certain format of fields, use MySQL

- Objectives: 1) To understand what form validation is
  2) To learn basic functioning of DOM objects
  3) To ~~learn~~ learn how to apply various techniques & implement

- Theory:

1) Explain role of regular expression. → Why are they suitable for validating formats like phone no., name, email or presence of specific char. in password)

→ Regex are a declarative way to define patterns of text. Instead of writing manual parsing logic (loops, conditions) regex lets us describe input should like in ~~is~~ rule-based

- Role of Regex

Pattern Matching: Quickly ~~s~~ checks if string follows specific structure
Search: Find substrings (like emails in text)
Replacement: Rewrite text that matches certain pattern
Validation: Confirm input meets constraints without writing verbose logic.

eg. `^/+?[0-9]{10,15}$` => Check phone no.

Q.2 Explain the fundamental diffrence between session & a cookie in web development. How they work together to main logged-in state).

→

	Cookie	Session
Storage	Client-side (browser)	Server-side (Memory, DB)
Data Stored	Small key-value pairs (eg. session-id, prefs.)	User related states (eg. user_id, roles, cart)
Security	Can be tampered unless protected (Http only, Secure)	More Secure (not exposed to client, only)
Size Limit	~4KB per cookie	Depends on server resources
Transmission	Sent automatically with every HTTP request	Accessed by server when session-id provided
Failure Case	If deleted by user, server can't identify	If cleared then cookie points nothing → logged out

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

→ Client side validation: Instant feed back, improves user experience and prevents obvious bad inputs from hitting server.

Server side validation: Enforces actual rules on the data that reaches the server. Since attackers can bypass the browser (disable javascript or use tools like CURL or postman), the server must verify the input.

Scenario: Suppose an e-commerce site only uses client side javascript to validate that price field of product can't be changed by user. A malicious user could disable JS & send an HTTP request directly to server, modifying price from $1 to $1000.

Q4 Provide simple eg. of how a javascript can interact with DOM to dynamically change content of web-page after user action like form submission.

→ HTML eg snippet for form:

```
<form id="myForm">
    <input type="text" id="username" required>
    <button type="submit">Submit</button>
</form>
<p id="great"></p>
```

Javascript eg.

```
Const form = document.getElementById("myform")
```

PAGE No.
DATE / /

```
form.addEventListener("submit", function (event) {
    event.preventDefault();
    document.getElementById("greet").textContent = "Hello " +
        document.getElementById("username");
```

User Submits Form → JS intercepts with event.preventDefault
→ Reads input value and update the <p> tag.

Q.5 Give steps for connectivity from frontend to MySQL using HTML, CSS, JS

→ 1) Create frontend using HTML, CSS, JS

2) Setup backend server (Node/Express, Flask, Spring)

3) Setup MySQL Server, create database and tables & insert some data

4) Connect backend to MySQL with connector libraries like mysql2, SQL Alchemy, Hibernate with secure DB connections

5) Implement CRUD API endpoints in backend

6) Call API from frontend javascript and send HTTP re to backend.

7) Parse the JSON response and update the DOM.

PAGE No.
DATE / /

* FAQ.

1) Write 3 reasons why form validation are important.

→ 1) Data integrity: Validation ensures that only proper and meaning ful data gets stored in the database, reducing cross & inconsistencies.

2) Security & Protection: Strong validation helps block malicious input like SQL-injection, cross-site scripting or attempt to manipulate hidden field, thereby protecting system from attacks.

3) Improved User experience: By providing immediate feedback on mistakes (eg. invalid email format), validation helps ~~users~~ users quickly correct errors and preventing failed submissions.

Q.2 Give an example on how to modify attribute value using dom.

→ HTML:

```
<p id="hello">hi</p>
```

JavaScript:

```
document.getElementById("hello").textContent = "hellooooo";
```

Q3 What are different features of JavaScript.

→ 1) Light-weight & Interpreted: Runs directly on browser without getting compiled

2) Object-Oriented:- Supports objects, interfaces and classes with inheritance for relationship

3) Event-driven: Can respond to user-clicks like key press and form submissions.

4) Platform independant:- Runs on any device/browser with JS engine

5) Community and Ecosystem:- Huge library and framework support (React, Angular, Vue, Express)

Conclusion: In this assignment, we learned how to manipulate DOM elements with JavaScript and validate user forms using regular expressions. Using Regex we validated phone numbers, emails and name.

DATE / /

Assignment -03

- Aim: Design interactive front-end application using React by implementing template using Components, St and props, class, events. It must be responsive to scal across different platforms.

- Objective: - Develop responsive interactive front-end applica using React.js that effectively demonstrates fundamental concept of component based architecture state mgmt and event handling. The app. will serve as practice exer in building scalable user interface by building templating with components, managing dynamic data with States and props & handling user interaction with event

- Theory

Q.1 Explain role of state and prop in React. How do they differ and what is primary purpose of each in managing data flow within compond based applicatio

→ In React, State is about managing data inside the component while props are about passing dat into a component. State is mutable and local while props are immutable and controlled externa whenever state is changed the whole component is re-rendered to reflect the new updated stat

while props are passed down from parent to child to configure or customize a child, they're immutable from child's perspective.

State
- Internal data storage, local to component
- Mutable ; updated with usestate hooks
- Component decides how & when it changes
- Used for dynamic values to form inputs, toggles

Props
- External input from parent to child
- Immutable, child can only read not write
- Controlled by parent, not by component itself
- Used for configuration, communication & reusability.

Q.2 What is React Component? Diffrence betⁿ class & fⁿ Component and discuss the advantages of using fⁿ components with hooks like useState and useEffect over class component.

A react component is a reusable piece of UI that takes inputs (props) and returns a React component describing what should appear on screen. Components lets you breakdown complex UIs into smaller manageable units.

PAGE NO.
DATE / /

Class Components

- Defined using class keyword & extends React.Component and must implement render() fn
- Uses lifecycle methods (componentDidMount, componentDidUpdate)
- State managed by this.state & updated by this.updateState

FC Components

- Defined as plain javascript object
- Received props as fn argument & return JSX
- Uses hooks like useState, useEffect, useContext.
- Simple & easier to test

Advantages of fn components

- Cleaner code: fn are shorter & avoid "this" binding.
- Hooks: Allows using hooks like useEffect & useState.
- Performance: Fn component are generally faster than
- Reusability:- Hooks allow to create custom hooks unlike lifecycle methods.

Q3 Describe the concept of "templating using components using React". Why is this considered superior to traditional web development methods that rely on monolithic HTML files?

PAGE No.
DATE / /

→ Templating using components in React means building UI by composing reusable, self-contained components instead of writing one large HTML file. Each component acts like template for part of the UI: it takes props as inputs, manages state as needed and renders UI elements. These components can be reused across pages & combined into bigger structure and updated independently making UI modular rather than monolithic.

✗ Advantages

- Reusability: Components can be reused in multiple places instead of copy-pasting chunks of HTML
- Maintainability: Smaller, focused components are easier to debug & update than editing single HTML file.
- Dynamic behaviour: Components can handle state & props allowing interactive UIs without spaghetti JS on top.
- Performance: React updates only parts of DOM that change using virtual DOM unlike static HTML reloads

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

PAGE No.
DATE / /

→ In react, user events (like clicks, input changes) are handled using event listeners written as props on JSX elements. Unlike plain HTML where you use inline attributes like onclick, React uses camelCase event names (like onClick) & expects a function reference

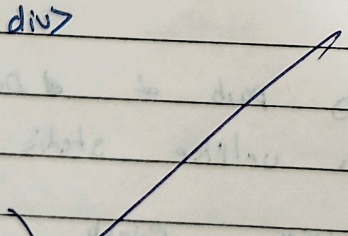
eg.

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

function Counter() {
  const [count, setCount] = useState(0);

  const handleClick = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p> Count: {count} </p>
      <button onClick={handleClick}> + </button>
    </div>
  );
}
```



Q.5 What is responsive web design & why is it crucial for modern applications? Describe how would you implement a ~~react~~ responsive react design using CSS media queries or CSS-in-JS library?

Responsive web design we build UIs that adapt fluidly to different screen size, orientation and input modes so the same app is usable on phones, tablets & desktops. That means using flexible layout primitives (flexbox, grid) and relative units (%, vw, rem) and break points so components reflow instead of breaking. In read we keep visual adaptation in CSS and reserve javascript for behaviour changes that truly need it.

eg. layout.css

- container { display: grid; grid-template-columns: 1fr }
- header { font-size: 1rem; }

@ media (min-width: 768px) {
  - Container { grid-templat-columns: 240px 1fr }
  - header { font-size : 1.125 rem }
}

@ media (min-width: 1200px) {
  - container { max-width: 1200px; margin: 0 auto;
  - header { font-size : 1.25 rem }
}

layout.jsx
import styles from "./layout.css"
export function Layout () {
return (
  <div className = { styles.container }>
    <aside> Side bar

);

Similarly CSS-in-JS example

theme.js

export const bp = { md: "768px", lg: "1200px" }

Card.jsx

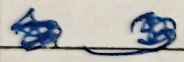
import styled from "styled-components"

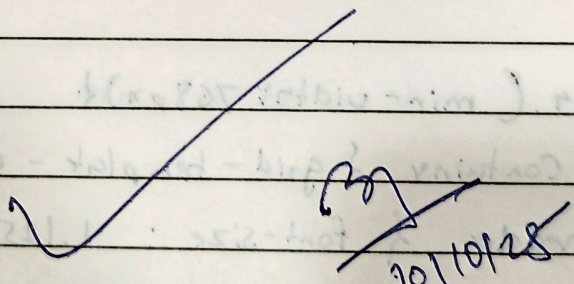
import { bp } from "./theme"

const Card = styled.div`

padding: 1rem;

@media (min-width: ${bp.md}) { padding: 1.5rem; }





30/10/25

Assignment -04

- Aim: Enhance web-page in earlier assignment by rendring list and portals, error handling & style with Reed CSS & make it responsive to scale well across PC, tablet, mobile
- Objective: 1) Enhance User interfaces and experiences
  2) Improve App. robustness & Navigation
- Theory

Q.1 How do Lists & keys work in React?

⇒ In React, Lists are just arrays of component elements rendered using .map().

eg.

```
const names = ["Yashraj", "Nakse", "Elon", "Mark"];

function NameList (){
    return (
        <ul>
        { names.map (name => <li>{name}</li>)}
        </ul>
    );
}
```

This renders list of names in <li> element.

Keys are unique identifiers attached to each element in list to help React identify which elements have changed, been added, or removed.

eg. {name.map(name => <li key={name}>{name})}

If keys are not unique, react falls back to index based keys which can cause bugs on dynamic lists.

Q.2 What is a React Portal, when do we use them?

→ A React Portal is a feature that allows to render a component's output to a DOM node that exists outside the parent component's DOM hierarchy.

It is created using: ReactDOM.createPortal(child, container) where child is the react element to render container is the target DOM node.

Eg.

```
ReactDOM.createPortal(
  <div> Hello </div>,
  document.getElementById("hello")
);
```

React portals are used for components like modal

PAGE NO.
DATE / /

tooltip, dropdown to visually break out of its parent container.

Discuss the importance of Error Boundaries in React.

→ Error Boundaries are special React components that catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI instead of crashing the entire application.

They are created using either (or both) of lifecycle methods:
static getDerivedStateFromError(err)
componentDidCatch(error, info)

eg

```
class ErrorBoundary extends React.Component {
  constructor(props){
    super(props);
    this.state = { hasError: false}
  }
  static getDerivedStateFromError(){
    this.state.hasE
    return { hasError: true};
  }
  componentDidCatch(err, info){
      console.error(err, info)
  }
```

render() {
return this.state.hasError ?
<h1>Something wrong?</h1> :
this.props.children
}
}

Importance: 1) Prevents entire app from crashing
2) Improves user experience
3) Simplifies Debugging
4) Isolates faulty component.

Q4 How does React Router enable Single Page Application (SPA) Functionality)

→ React Router enables SPA functionality by allowing navigation between different components or pages without reloading the entire ~~webpage~~ webpage.
It uses the browser's History API (pushState, popState) to update the URL & render different components dynamically, while keeping application running.

eg.

import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {

```
return (
    <BrowserRouter>
        <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/hello" element={<Hello />} />
        </Routes>
    </BrowserRouter>
);
```

This enables client side rendering, keeps smooth User interface & state persistence across components

Q.5 Explain Different ways to style a React App?

→ 1) Inline Styling

```
function App() {
    ~~<h1 style="color: blue">Hi</h1>~~
    <h1 style={{ color: "blue" }}>Hi</h1>
```

2) CSS Stylesheets

```
import "./App.css";
```

Simple & easy, global by default

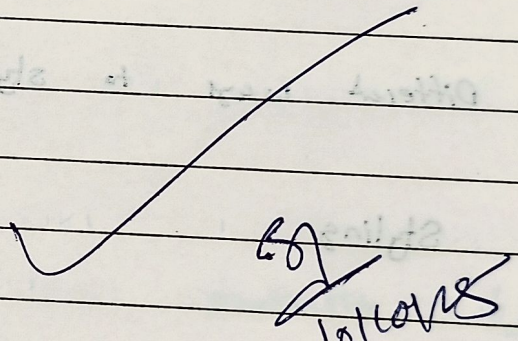
3) CSS Modules

Locally scoped CSS to avoid naming conflicts.

import styles from "./App.module.css";
function App() {
return (<h1 className={styles.heading}>Hi</h1>);
}

5) UI Frameworks: UI Frameworks like Bootstrap, Material UI for pre-built styled components.

Conclusion: Through this assignment we implemented React feature to enhance functionality & responsiveness improving user experience.



10/10/25

PAGE No.
DATE / /

Assignment -05

- Aim: Deploy a responsive web design using Express.js framework to perform CRUD operations and deploy with Node JS, use MongoDB.

- Objectives: 1) Develop a full stack web application
  2) Demonstrate Backend development & Deployment proficiency.

- Theory:

Q1 What is role of express.js as web framework for Node.js?

→ Express.js is a minimal & flexible web framework built on top of node.js that simplifies the process of building server-side ~~app~~ applications and APIs.

It provides high-level abstraction over Node's core 'http' module, making it easier to handle routing, middleware, request & response.

Express.js supports.

1) Routing: Handles different URLs & HTTP methods (GET, POST).

app.get('/user', (req, res) => res.send("Hi!!"));

2) Middleware Support:
Allows use of function to process request (eg. logging, auth)

app.use(logger.log());

3) Simplified Setup:
Setup & spin web-server easily
app.listen (8012);

4) Integration Friendly: Works with MySQL, Mongodb, and with Read.

Q.2 Explain concept of CRUD in context of web application.

→ CRUD in the context of web development stands for CREATE, READ, UPDATE, DELETE. which are four basic operations. used to manage data in a web application.

1) CREATE

Means Adding new data to the database, usually with an post request
eg. POST /user

2) READ

Retrieving some existing data from Database, usually with GET request.

eg. GET /user/1 to get user with id=1

3) Update

Modify some existing data in database, usually with PATCH / PUT requests.

eg. PUT /user/1 to edit user with id=1

4) Delete

Remove data from database, usually with DELETE request

DELETE /user/1 to delete user with id=1

These operations correspond how a backend (like Express + database) handles client systems & maintain data integrity.

Q.3 Why is MongoDB suitable for this Project?

→ MongoDB is suitable choice because of its NoSQL design choice with flexibility, scalability & high performance making it ideal for web applications.

It stores data in NoSQL JSON-like documents (BSON) instead of tables which allow for easy dynamic schema for unstructured data, perfect for data formats where format changes frequently.

PAGE No.
DATE / /

Feature of MongoDB:

1) Schema-less Structure: Store varying types of data
2) High-Scalability: Supports horizontal scaling through sharding
3) Fast-Performance: Uses in-memory caching & fast indexes for quick
4) Built in JSON support:- Simple & easy to retrieve data.

Q4 What steps are involved in deploying a Node application?

-> Development typically involves preparing project for production, setting up environment variables, connecting a database and hosting ~~platform~~ on a platform like Vercel, Render, Netlify etc. We can also use cloud platforms like AWS EC2 or Azure for deploying. We need to configure & secrets to make the project work. We need to ensure MongoDB is reachable & routes are displayed with correct configuration. We need to automate & create CI/CD pipeline.

Conclusion: In this experiment, we developed a full stack app using Node.js, Express and MongoDB. This helped us understand backend development, database integration & deployment for responsive data-driven web systems.