Day wise Curriculum

Day 1

1. Full stack Breakdown - Why MERN stack
2. Frontend vs Backend. 4 months roadmap
3. Explain client-server architecture. Explain role of each technology involved in MERN
4. **VS Code BASICS**
   1. Installing VS Code.
   2. Important VS Code extensions.
   3. Basic VS Code features (file navigation, debugging, integrated terminal)
5. Intro to HTML, CSS
6. Run HTML in browser using VS code

Task -

1. Setup **VS code**
2. List **useful vs code extensions** for web development.
3. Do research on **Browser Development Tools**
4. List **2 famous applications** built using MERN and inspect the application.

HTML

Day 2

1. How Does the Internet Work? How url works when we search anything in google.
2. DNS look-up, Components of a URL and hosting
3. Types of websites
4. Html intro-, html5
5. Structure of html head ,body
6. Parts of HTML element, starting tag, closing tag, attributes and content
7. Heading tags -> h1 to h6
8. What is the difference between h1 to h6
9. Intro of p tag
10. Div tag - Semantic vs Non-Semantic tags
11. ul, ol & List tag
12. Img tag, src,alt attribute
13. Button
14. Span, strong, b, u, i tags
15. Br
16. Hr

Task -

1. Create your own static "**Resume**"..Include all the tags that we have learnt.

Day 3

1. Anchor
2. Iframe
3. Form Tag with 14 different input methods
4. Meta tags theory
5. Title tag and Link tag for adding FAVICON for the tabs.
6. Relative paths and absolute paths and its difference. while adding the images with img tag and links with a tag
7. Adding the symbols with metadata, it should be a practical explanation.
8. Inline and block level elements in the HTML.
9. Tables, colspan, rowspan in tables.

Task -

1. Continue with the "**Resume**" to implement what we have discussed..
2. Create a Form using HTML.

CSS

Day 4

1. Intro to CSS
2. CSS rule sets, Basic styles (like font, colour, bg color)
3. Inline CSS, Internal CSS, external CSS
4. Types of selectors - Class, ID, Tag, universal selectors
5. CSS specificity
6. Important keyword
7. CSS inheritance

Task -

1. Research on how we debug elements, styles - dev tools
2. Create a **card** using class and ID selectors.

Day 5

1. Colors
2. Typography
3. Text Styles
4. CSS Box Model (Padding, Border, Margin, Height & Width)
5. Developer Tools

6. Pseudo Selectors, Pseudo classes
7. Priority Selectors

Task -

1. Add colors to the card.
2. Add colors to the resume
3. Add typography to the resume

Day 6

1. Display - flex, grid, inline, block, none
2. Flex Box and flex properties
3. CSS Grid and grid properties
4. Navbar design

Task -

1. Create a **responsive Navbar** using **Flexbox or Grid**.

Day 7

1. Diff between em, rem, px, width, min-width, max-width, height, min-height and max-height, vh, vw, %
2. Background Image stylings
3. Gradient
4. Opacity
5. Text shadow & Box Shadow

Task -

1. Create a **glowing button effect** using shadows..in the nav bar

Day 8

1. Positional Properties(Static, Absolute, Relative, fixed and sticky)
2. Examples of those positional properties
3. CSS variables
4. Border box

Task -

1. Create a **theme switcher** using CSS variables.

Day 9

1. OverFlow properties

2. Develop a simple static page of a real time website (Swiggy/Zomato/Netflix/Flipkart etc.)
3. Responsive Design- CSS Media-queries
4. CSS animations and Transition effect.

Task -

1. Create a **loader animation** using only CSS.
2. Use mediaQueries To "**Resume**"

Project : Create a multi page portfolio - using HTML, CSS, Add animations, responsiveness.

Bootstrap & UI-project

Day 10

1. Intro to Bootstrap CDN
2. Component Section
3. Bootstrap Class names

Task -

1. Build a **Bootstrap landing page** with a navbar, hero section, and a call-to-action button.
2. Create Cards using Bootstrap, ensuring no prebuilt components are used.

Day 11

1. Continuation of Component Section
2. Utility Section
3. Layout Section
4. Bootstrap Grid system
5. Break Points

Task :

1. Create a **responsive Bootstrap portfolio page** using the grid system & utility classes.

Day 12

1. Carousels
2. Modals
3. **GIT BASICS**

   1. Git concepts - Intro, Usecases, Advantages, Setting up Git,
   2. Git commands - (init, clone, add, commit, push, pull, status, log)

3. Git branching - (branch,
4. **Deployment** using Netlify/Vercel
5. Explain Major Project

Task :

1. Install and configure GIT on your PC. Push any of your code to a remote repo

Day (13, 14, 15, 16)

1st Major Project - Using HTML, CSS & BootStrap (Like. Ecommerce Site.).
Build an E-commerce Website using an API link provided.

Day 16

1. Doubts clearing
2. Revision

Task :

1. Deploy a **multi-page project (previously built projects)** using Netlify.

Tasks - Create at least 5 different website design using html, css, bootstrap)
(Max up to 18 days all the above has to finished)

JavaScript

Day 17

1. Intro to JS
2. diff b/w scripting vs programming
3. how to add to html file
4. Data types
5. Primitive Data Types(String, Boolean, Number, undefined, null, BigInt)

Task :

1. **Add JavaScript into your HTML file and run it in the browser.**
2. Write a **JavaScript program** that declares and logs a variable of each data type.

Day 18-19

1. Continuation of Primitive data types
2. Non-primitive data types (Date, Array, Arrays)
3. Operators -Arithmetic operators, Comparison operators, Logical operators, Ternary operators, Assignment Operators
4. Truthy vs falsy values
5. Type Coercions (Implicit and Explicit).
6. Control statements - Conditional statements(if, if-else, else-if-ladder, switch-case) + loop statements (for, while, do-while)
7. Different ways of iterations
8. Types of errors - Reference errors, Syntax errors, Logical errors
9. Problems

Problems :

Finds the **largest of three numbers** using an if-else statement.

Uses a **switch-case** to print the **day of the week** based on user input.

Prints the **sum of numbers from 1 to N** using a loop.

Day 20

1. For of loop, for in loop
2. Continue and break
3. Problems on loops, if-else

Task :

1. Create a **simple JavaScript calculator** using operators & control statements.

Day 21

1. Intro to Data Structure (Arrays) - CRUD with Arrays
2. Array methods - push, pop, shift, unshift, sort, split, slice, splice, indexOf etc.
3. Loops with Arrays - for in, for of
4. Nested arrays - Matrix sum
5. Problems on Arrays

Problem :

Finding the **largest number** in an array.

Removing **duplicate elements** from an array.

Reversing an array **without using reverse() method**.

Merging two arrays **without duplicates**.

Task :

1. Create a **To-Do List App** using arrays & CRUD operations.

Day 22

1. Intro to Data Structure (Objects) - CRUD with Objects
2. Dot notation vs Bracket notation
3. Object methods - object.keys, object.values, object.entries, etc.
4. Nested Objects
5. Problems on Objects

Write programs for:

- Finding the **total price of products** from an array of objects.
- Merging two objects **without overwriting existing keys**.
- Counting the number of **occurrences of each word** in a sentence.
- Creating a **student record system** using objects.

Task : Create a **user profile system** that stores multiple users as objects inside an array.

Day 23-24

1. Composite Data Structures - Array of Objects - CRUD
2. Array destructuring, Object destructuring
3. Pass by value and Pass by reference - Examples
4. Spread Operator
5. Shallow copy vs Deep copy
6. Problems

Problem :
Sorting an array of objects based on a property (e.g., age, price).
Finding a specific object in an array based on a key.
Creating a copy of an object without modifying the original.
Merging two arrays of objects based on a unique key.

Task : Create a simple inventory management system using an array of objects, allowing users to add, update, delete, and search items.

Day 25-26

1. Functions - advantages

2. Arguments and Params, Default parameters
3. Rest operator
4. Function statement
5. Function Declaration
6. Named function
7. Function Expression
8. Named function expression
9. Anonymous function
10. Immediate function
11. Arrow functions
12. First-class functions
13. Problems

Problems :

A function to **find the maximum of three numbers**.

A function that **returns the factorial** of a number.

A function that **returns the sum of all numbers** passed as arguments.

Convert a **normal function** into an **arrow function**.

Create an **IIFE** that prints today's date.

Task :

1. Create a **higher-order function** that takes a function as an argument and applies it to an array of numbers.

Day 27

1. Hoisting - var, function declaration
2. Var, let and const differences
3. Block Scope, functional and global and lexical scope.
4. Scope and Visibility
5. Reference Error, undefined, not defined

Problems :

Demonstrating **hoisting** with var, let, and function declarations.

Creating a function that returns a **closure**.

Demonstrating **block scope vs function scope**.

Handling **ReferenceError and undefined variables**.

Task : Implement a **counter function** using closures.

Day 28

1. Callback functions basic
2. Intro to higher Order functions
3. Map function
4. Reduce Function
5. Filter Function
6. For each Function

Problem :

Creating a **custom map function** using forEach().

Using reduce() to **find the maximum number** in an array.

Filtering out **students with scores above 75** using filter().

Using map() to **capitalize names** in an array.

Day 29

1. String methods
2. ASCII
3. Temp literals
4. Practical Problems related to this that include
   1. Finding prime number
   2. Palindrome
   3. Fibonacci Series
   4. Anagrams.

Day 30-31-32

1. Window object, Methods - window.location, reload, href etc.
2. HTML DOM, DOM tree, nodes
3. DOM manipulation with JS
4. Accessing and modifying HTML elements with JS - getElement by ID/class/tag,textContent, appendChild, querySelector, querySelectorAll, innerText, innerHTML, createElement, setAttribute
5. Modifying CSS styles with JS - classList.add, classList.remove, style
6. Event, Event types, Event listeners - through addEventListener, HTML in-line
7. Event propagation - Event Bubbling, Event Capturing
8. Event Handling in JS

Problems :

Write a function that changes the **background color** of a div when a button is clicked.

Create a form that captures user input and **displays the entered data dynamically**.
Implement a **to-do list** where users can **add and remove tasks dynamically**.
Demonstrate **event bubbling** with multiple nested elements.
Create a **modal popup** that appears on button click and disappears on another button click.
Use addEventListener() to detect different types of events (click, input, mouseover).

Day 33

1. Adding Cards dynamically to DOM through JS
2. Basics of Storage.
3. Client-side storage - Local Storage, Session Storage and Cache Storage
4. Use cases of Local storage in the website
5. JSON Object and its theory.
6. JSON.stringify
7. JSON.parse
8. Persisting with reload using Client side storage mechanisms

Problems :

1. Store a user's theme preference (light/dark mode) in Local Storage.
2. Use Session Storage to store a temporary form input value.
3. Save an array of objects in Local Storage and retrieve it later.
4. Create a simple to-do list where tasks are stored in Local Storage.
5. Convert a JavaScript object to JSON and store it, then retrieve and display it.

2nd Major Project - Using HTML, CSS, Bootstrap & Javascript
(Design a Calculator using DOM in JS / Adding Cards dynamically to DOM through JS)

Day 34-35

1. Intro to Class Object
2. This object, this context in different scenarios
3. New operator, Constructor functions
4. Class instance creation - Properties and Methods
5. Oops - Inheritance, abstraction, polymorphism, method overriding
6. __proto__ object, prototypal inheritance
7. Practical problems - Car, Bank. Animal examples

Task : Write a Car class with specified methods. Create an object from the Car class and test the methods in the console.

Day 36

Revision of all the above concepts.(Like a crash course within a couple of hours, Instructors have to recap the concepts of hoisting, functions, array, object, loops, call by reference and call by value with simple programs and should explain everything once again comparatively. It's like a doubt handling session and so has to address the student's doubts as well.)

Async JS

Day 37

1. Intro to Async programming
2. setTimeout, setInterval, clearTimeout, clearInterval
3. Javascript Engine Explanation (theory part)
4. Callback functions in async programming
5. Callback hell or Pyramid of doom

Problems :

Create a countdown timer using setTimeout.

Write a program that **changes background color every 2 seconds** using setInterval.

Demonstrate **callback functions** using a function that simulates **fetching user data.**

Write an example where **callback hell occurs**, then refactor it for better readability.

Task :

Implement a **stopwatch** with start, pause, and reset using setInterval and clearInterval.

Day 38

1. Promise Object
2. Promise states
3. Resolving promises, error handling - .then, .catch
4. Async/await
5. Error handling - Try catch

Problems :

Create a function that simulates API fetching using Promises and resolve/reject it randomly.

Convert a callback-based function to use Promises.

Use Promise chaining to execute three async tasks in sequence.

Use Promise.all() to fetch data from three different fake APIs.

Use Promise.race() to display the fastest response from two API calls.

Task : Implement a **"loading"** while waiting for a Promise to resolve.

Day 39

1. How does JS handle the asynchronous process?
2. Event loop - Call stack, micro-tasks, macro tasks
3. Visualization of event loop
4. Promise.all(), Promise.race()
5. Promise Chaining

Problem :
Convert a function using Promises to async/await.
Create a fake API call using async/await and handle errors properly.
Use setTimeout and a Promise together to test event loop behavior.
Write an example that demonstrates the difference between microtasks and macrotasks.
Visualize and explain the event loop execution order of the following code:

Task : **demonstrates the event loop in action**.

Day 40

1. Fetch method of handling Apis (only basic level)
2. Rendering HTML through APIs - fetch API (only GET)
3. Using Client storage mechanism for persisting API data
4. Adding custom data through forms to create cards

Problems

1. Make an API request using fetch() to get a random user from an API and display their name on the webpage.
2. Use fetch() to retrieve an image from an API and display it dynamically.
3. Handle errors gracefully when the API request fails

Advanced JS concepts

Day 41

1. Currying
2. Closures
3. Call
4. Bind
5. Apply
6. jQuery basics
7. Major Project 3 Explanation

Problems

Convert a normal function to a curried function.
Create a function that returns a closure and maintains a private variable.

Use call, apply, and bind to invoke a function with different values.

Write a function that uses closures to create a counter that increments, decrements, and resets.

Demonstrate the difference between call, apply, and bind with an example.

Buffer of 4-5 days can be given so far and after this heading successfully to React

Day 42-43-44-45

3rd Major Project - Using HTML, CSS, Bootstrap & Javascript.

React JS

Day 46 (UI—-LAYER)

1. Intro to react js
2. Advantages, Why React.js?
3. Single page applications (SPA)
4. Virtual DOM
5. Ways to install React.js

Problem :

1. Explain the advantages of React.js in your own words.
2. What is an SPA? Give an example of a popular website that follows the SPA architecture.
3. Describe the difference between the Virtual DOM and Real DOM.
4. Set up a simple React project using create-react-app or Vite and run it.
5. Create a basic React component and render it inside App.js.

Day 47

1. Node setup
2. How to use NPM?
3. How to create package.json and purpose of it? Purpose of node_modules & npm install
4. Create and run a Basic App with React JS (CRA & Vite) and other Supported NPMs
5. Entry points - CRA vs Vite.
6. Explanation of basic folder structure of react. Comparison between CRA vs Vite app. Single Page rendering

Problems

1. Install Node.js & verify the installation using terminal commands.
2. Create a package.json file and explain its purpose in your own words.
3. Install a new package (e.g., lodash) using NPM and import it in a JavaScript file.
4. Create & run a basic React app using both CRA & Vite, then compare their folder structures.
5. Explain the difference between node_modules and package.json.

Day 48

1. How to use Named import/exports and Default import/exports in React
2. Explain React components, Functional vs Class Components - reusability
3. Explain JSX and Embedding JS expressions in JSX
4. Create React components with React.createElement. Compare it with JSX
5. Mechanism of how react renders JSX to DOM
6. Nested Components

Task : Create a **Navigation Component** with multiple links using Nested Components

Day 49

1. Intro to Class Components
2. Explain Constructor function, local state management, render method and adding custom methods in class based components
3. Updating state with this.setState() and re-rendering of components
4. Small counter app with Class components

Task : counter app

Day 50

1. Passing props in Class components, destructuring of Props
2. Conditional rendering & Infinite rendering
3. Pure functions, Side effects on React
4. React lifecycle methods in Class components

Task : Create a **Timer Component** that starts when mounted and stops when unmounted.

Day 51

1. Intro to Functional Components.
2. Explain local state management with React useState Hook
3. Small counter app with Function component
4. Passing props in Function components, destructuring of Props
5. Props from Child to parent and from parent to child

Task : Modify the counter app to **increment/decrement by a user-input value**.

Day 52

1. Intro to inline, internal, and external CSS in react
2. Intro to React Bootstrap
3. Static webpage using React Bootstrap
4. Intro to **Tailwind CSS**

5. Other UI libraries - Radix UI, Material UI etc.

Task : Create a **responsive navbar** using **React Bootstrap** and **Tailwind CSS**.

Day 53 (FUNCTIONAL LAYER)

1. Basic toggle program using both the functional and class components.
2. UseEffect Hook
3. Explanation of React lifecycle method in functional component.
4. Difference between props and setState in class components.

Task : Implement a **Dark Mode Toggle** using useState & useEffect.

Day 54

1. Intro to React Router Dom
2. Explain Router, Routes, Switch
3. Create a simple website using react-router DOM
4. Explain Dynamic Routes
5. Navigation, Link tag with react-router-dom

   Task : Build a Multi-Page React App with React Router DOM

Day 55 (DATA LAYER)

1. Intro to Axios and http methods.
2. Basic program to explain the http methods using Axios.
3. Handling data using state
4. Comparing axios and fetch.
5. Intro to Async programming.(async/await in react js). Error Handling with try,catch

Task :

Build a Simple To-Do App with Axios

1. Create a React app that fetches To-Do items from an API (https://jsonplaceholder.typicode.com/todos).
2. Display a list of first 10 To-Dos in state.
3. Add a button to delete a To-Do (DELETE request).
4. Use an input box to add a new To-Do (POST request).
5. Use async/await for API calls

Day 56

1. Using Higher order functions in React - filter, map, reduce array data
2. Imperative vs Declarative programming
3. Prop drilling of data

4. Methods to avoid props drilling
5. Global State management and basics of useContext hook

Task : Build a Product List with Filtering & Global State

Day 57-65

1. Making a frontend e-commerce/medi-assist website that covers major react concepts like React hooks, React-Router DOM, axios, async, useContext, and higher-order functions like map, filter etc.
2. UI libraries like React Bootstrap, Mui, Radix UI, and Formik library will be explored
3. Website will include features like
    - Login feature (with localStorage + useContext hook),
    - Landing page built with diverse UI libraries
    - Rendering Dynamic Cards with API data using axios
    - Search and Filter functionality for cards
    - Item details page (Opened when clicked on Cards)
    - Add to Cart feature and add/remove quantity feature
    - Persisting Cart details & login using localStorage

Day 66

1. Need of Redux store
2. Intro to Store
3. Need of Store provider
4. Sample counter program using redux

Day 67 (OPTIMIZATION LAYER)

1. React optimization techniques
2. UseMemo Hook and useCallBack
3. Lazy loading techniques.
4. Avoid unnecessary renders
5. Higher Order Components
6. Custom Hook
7. Code splitting

Day 68-72

4th Major Project - Using HTML, CSS, Bootstrap, Javascript & React JS (FRONTEND CAPSTONE PROJECT) Front end project like Netflix, amazon or swiggy clone

NodeJS

Day 73

1. Introduction to Node.js - What is Node Js, V8 Engine
2. Why Node js - Explain the advantages
3. Intro to Node.js
4. CJS - Import and exports in Node.js
5. Inbuilt modules, internal modules and external modules (path, fs, os, events, http)
6. Node REPL

ExpressJS

Day 74

1. Creating a server with node.js using http in-built module
2. Running a server with node.js and sending a response
3. Creating package.json, node_modules
4. Intro to Express
5. Creating Server and GET API using Express.js
6. Intro to Nodemon

Day 75-76

1. Intro to RESTful APIs
2. Principles of REST API
3. HTTP Basics & API Communication
4. HTTP methods (GET, POST, PUT, DELETE), status codes, Request headers and response headers
5. Query parameters vs. Path parameters
6. CRUD operations using REST APIs
7. Intro to Postman (API testing)
8. Testing using Postman

Day 77

1. CRUD operations with APIs using frontend
2. Implementing MVC Architecture in Node.js
3. Routing with Express
4. Creation of Controllers
5. Handling errors for all API endpoints
6. Error responses with appropriate status code

MongoDB

Day 78

1. Intro to Databases. DBMS (Database Management systems)
2. Relational vs Non-Relational DBMS. Comparison
3. MongoDB Account and Account setup using mongoose
4. Intro to dotenv package
5. Connect Node application with MongoDB.
6. Intro Schemas of Databases
7. Model Creation and Validation with Mongoose

Day 79

1. CRUD Operations with MongoDB methods
2. Creating a Full-stack app. Integrating Frontend + Backend (Login, Signup, Other APIs)
3. Error Handling
4. Intro to bcrypt for password encryption
5. Intro to Middlewares
6. Authentication vs Authorization

Day 80

1. Intro to JSON Web Token(JWT) and JWT authentication mechanism
2. Authorization Headers
3. Generating JWT Tokens & Secret Keys
4. Creating Middleware function with JWT authentication (Sign in & Verify)
5. Integrating middleware with API endpoints
6. Integrating JWT auth with existing frontend.
7. Error Handling

Day 81-90

Final Full Stack Project - Using HTML, CSS , Bootstrap, Javascript, React Js, Node JS & MongoDB

- **LIKE: CAPSTONE FULL STACK PROJECT LIKE EITHER AMAZON, SWIGGY OR NETFLIX OR ANY PRACTICAL APP CLONE.**