Training Syllabus
[2 years Duration]

Frontend Development with JavaScript

Join us for Better Career and Job Guarantee

[Live Online & Offline Classes Available]

LEARN-2-EARN LABS TRAINING INSTITUTE, AGRA

Website: www.LearntoearnLabs.com | Contact : +91-9548868337

Frontend Development Training Program (Two Years Duration)

The **Front-End Development Training Program** at Learn2Earn Labs Training Institute is designed to equip aspiring developers with the skills and knowledge required to build interactive, high-performance, and responsive web applications. This program provides in-depth training in front-end technologies, including HTML, CSS, JavaScript, TypeScript, React.js, Next.js, Vue.js, Angular, Redux, UI Design, and DevOps basics.

This comprehensive program is crafted to transform beginners into professional front-end developers, empowering them to build real-world applications using modern frameworks, tools, and best practices.

About the Program

Our Front-End Development Training Program offers a structured learning path that takes students from fundamentals to advanced front-end technologies. With a curriculum designed to cover essential topics such as HTML, CSS, JavaScript, TypeScript, UI/UX best practices, state management, API integration, testing, and performance optimization, students will gain a strong foundation in building scalable and user-friendly applications.

This program includes practical hands-on projects, live coding sessions, and real-world case studies, ensuring students can apply their knowledge effectively in professional front-end development environments.

Who is a Front-End Developer?

A Front-End Developer is responsible for designing and implementing the user interface (UI) and user experience (UX) of web applications. A successful front-end developer possesses expertise in:

- Responsive Web Design (RWD)
- Component-Based UI Development (React, Vue, Angular)
- API Integration & Performance Optimization
- Cross-Browser Compatibility
- UI Testing & Debugging
- Modern CSS Frameworks (Tailwind CSS, Bootstrap, Material UI, Sass)
- Version Control (Git & GitHub)
- Deployment & DevOps Basics

By mastering front-end technologies, developers play a crucial role in **enhancing the user experience and ensuring seamless interactions** on web platforms.

What Makes This Program Unique?

- **Project-Based Learning** – Work on real-world projects and build a strong portfolio.
- **Industry-Relevant Curriculum** – Learn modern front-end development practices aligned with industry standards.
- **Comprehensive Syllabus** – Covers everything from basics to advanced front-end frameworks and state management.
- **Hands-On Training** – Focus on practical implementation rather than just theoretical concepts.
- **Expert-Led Sessions** – Gain insights from experienced industry professionals.
- **Mentorship & Guidance** – Get personalized mentorship to enhance your learning experience.
- **Job Readiness** – Learn the skills that top companies look for in front-end developers.

By the end of the program, you will be fully equipped to take on front-end development roles in startups, tech companies, or as a freelance developer.

Career Options After Completion

Upon successfully completing this Front-End Development Training Program, you can pursue roles such as:

- Front-End Developer
- UI/UX Developer
- React.js Developer
- Vue.js Developer
- Angular Developer
- JavaScript Developer
- Web Developer
- Front-End Engineer
- Mobile UI Developer (React Native)
- Full Stack Developer (after learning backend technologies)

With the growing demand for front-end developers, this program provides a gateway to high-paying career opportunities in the IT industry.

Who Can Join This Program?

This program is ideal for:

- **Beginners & Students** – Anyone new to web development who wants to learn front-end technologies from scratch.
- **Aspiring Developers** – Individuals looking to start a career as a front-end developer.
- **Software Engineers & IT Professionals** – Developers who want to specialize in front-end development and enhance their skills.

- **Freelancers & Entrepreneurs** – Those who want to build their own websites and web applications.
- **Designers & UI/UX Enthusiasts** – Professionals with basic knowledge of web design looking to expand their technical skills.
- **Tech Enthusiasts** – Anyone passionate about frontend development and ui-ux design can enroll.

No prior experience in coding or programming is required—just a passion for learning and a commitment to mastering front-end development.

Advantages of the Program

- **Learn from Scratch** – This program is beginner-friendly and progresses to advanced levels.
- **100% Practical Training** – Focus on hands-on coding, real-world projects, and industry applications.
- **Expert-Led Mentorship** – Learn from experienced front-end developers and industry professionals.
- **Live Projects & Case Studies** – Work on live projects and case studies for real-world exposure.
- **Portfolio Development** – Build a strong project portfolio to showcase your skills to employers.
- **Interview Preparation** – Get guidance on resume building, interview questions, and coding assessments.
- **Internship & Placement Support** – Gain internship opportunities and job placement assistance.
- **Global Opportunities** – Work remotely for international companies or launch your own frontend projects.

By the end of the program, you will be proficient in building modern, interactive, and scalable frontend of web applications.

How Much Dedication is Required?

This is an intensive training program that requires:

- **Commitment & Consistency** – Attend regular sessions, complete assignments, and work on projects.
- **Hands-On Practice** – Coding practice is essential to mastering front-end development.
- **Self-Learning & Exploration** – Stay updated with new front-end technologies and industry trends.
- **Project Work** – You will be required to develop real-world projects to enhance your problem-solving skills.

With dedication and hard work, this program will help you become a job-ready front-end developer.

Future Scope

The demand for front-end developers is steadily growing as companies continue to build digital platforms. After completing this program, you can:

- Work as a front-end developer in leading tech companies.
- Pursue a career in UI/UX development.
- Become a React, Vue, or Angular developer.
- Transition into full-stack development by learning backend technologies.
- Start freelancing or remote development work.
- Build personal projects, startups, or e-commerce platforms.

With continuous learning and skill improvement, you can grow into senior roles such as Front-End Architect, UI/UX Lead, or Technical Consultant.

Work Experience to Boost Your Career

At Learn2Earn Labs Training Institute, we go beyond theoretical training and provide:

- **Real-World Projects** – Gain practical work experience with hands-on projects.
- **Internship Opportunities** – Eligible students get internship experience to enhance their resume.
- **Portfolio Development** – Showcase your projects and skills to potential employers.
- **Industry Collaboration** – Work on live case studies and projects with industry partners.

Our structured approach ensures that you not only learn but also apply your knowledge in real-world scenarios.

Transform Your Career Today

If you're **passionate about front-end development**, this training program is your gateway to a successful tech career.

🚀 **Your Journey to Success Starts Here!**

🔥 **Learn. Code. Build. Grow.** 🔥

Training Content (Syllabus)

HTML

Module 1 – HTML Basics : Introduction to HTML, History and Evolution of HTML, Features of HTML5, Basic Structure of an HTML Document, Setting up the Environment: Text Editors (VS Code, Sublime Text, etc.), Web Browsers; HTML Basics : HTML Elements and Tags, HTML Document Structure, <html>, <head>, <body>, HTML Headings: <h1> to <h6>, HTML Paragraphs: <p>, Line Breaks and Horizontal Lines: <br> and <hr>, HTML Comments.

Module 2 – HTML Core Concepts : HTML Text Formatting : Bold, Italic, Underline: <b>, <i>, <u>, Strong and Emphasis: <strong>, <em>, Subscript and Superscript: <sub>, <sup>, Preformatted Text: <pre>, Mark, Small, Delete, Insert: <mark>, <small>, <del>, <ins>, Blockquote and Citation: <blockquote>, <cite>; HTML Lists : Ordered List: <ol>, Unordered List: <ul>, Nested Lists, Description List: <dl>, <dt>, <dd>; HTML Links : Creating Links: <a>, Absolute vs. Relative URLs, Linking to External Websites : Linking to Sections on the Same Page (Anchor Links), Opening Links in New Tabs: target="_blank", Adding Download Links; HTML Images : Inserting Images: <img>, Image Attributes: src, alt, title, width, height, Responsive Images with srcset, Image Maps: <map> and <area>, Optimizing Images for Performance; HTML Tables : Basic Table Structure: <table>, <tr>, <td>, <th>, Table Attributes: colspan, rowspan, Adding Borders and Spacing, Styling Tables with CSS, Responsive Tables; HTML Forms : Form Structure: <form>, Input Elements:, Text Input: <input type="text">, Password Input: <input type="password">, Radio Buttons: <input type="radio">, Checkboxes: <input type="checkbox">, File Upload: <input type="file">, Form Labels and Accessibility: <label>, Dropdowns: <select> and <option>, Textareas: <textarea>, Buttons: <button>, Submit and Reset, Modern Input Types:, email, number, date, time, tel, color, range, Form Validation: required, pattern, and Custom Validations, Grouping Form Elements: <fieldset> and <legend>.

Module 3 – HTML Advanced Concepts : Multimedia in HTML : Adding Audio: <audio> Tag, Supported Formats: MP3, OGG, WAV, Adding Video: <video> Tag : Attributes: controls, autoplay, loop, muted, Supported Formats: MP4, WebM, OGG, Embedding YouTube, Using the <embed> and <object> Tags; HTML Semantic Elements : Introduction to Semantics, Header and Footer: <header>, <footer>, Navigation: <nav>, Main Content: <main>, Articles and Sections: <article>, <section>, Aside and Figures: <aside>, <figure>, <figcaption>, Time and Address: <time>, <address>; HTML Graphics and Canvas : SVG Basics (Creating Shapes: <circle>, <rect>, <line>, etc., Adding Inline SVG Graphics), Introduction to Canvas: <canvas>, Drawing Shapes and Lines, Adding Colors and Gradients, Simple Animations; HTML5 APIs : Geolocation API, Drag and Drop API, Web Storage: Local Storage and Session

Storage, Browser Cache and Application Cache; Introduction to Web Workers, HTML5 Offline Mode.

Module 4 - Intermediate Topics : Responsive Design and Meta Tags : Viewport Meta Tag for Mobile Optimization, Responsive Images and Videos, Importance of Accessibility and ARIA; SEO and HTML : Meta Tags for SEO: <meta>, Description, Keywords, Robots, Adding Social Media Cards, Open Graph (Facebook) and Twitter Cards, Sitemap Creation and Importance, HTML Best Practices: Writing Clean and Semantic Code, Accessibility Guidelines (WCAG), Use of ARIA Roles and Attributes; HTML5 Latest Features: Native Lazy Loading: loading="lazy", Picture Element: <picture> for Responsive Images, Improved Input Types and Validation, Future of HTML.

CSS

Module 1 – CSS Basics : Introduction to CSS, History and Evolution of CSS (CSS1, CSS2, CSS3), Advantages of CSS, CSS Syntax, Types of CSS: Inline CSS, Internal CSS, External CSS; Connecting CSS to HTML: <link> Tag, style Attribute; CSS Basics : Selectors (Universal Selector (*), Element Selector, Class Selector (.), ID Selector (#), Grouping Selector), Combining Selectors (Descendant Selector, Child Selector (>), Adjacent Sibling Selector (+), General Sibling Selector (~)), Pseudo Classes & Pseudo Elements, CSS Comments, Inheritance and Specificity, Units in CSS: Absolute Units (px, cm, mm, in), Relative Units (%, em, rem, vw, vh).

Module 2 – CSS Core Concepts : CSS Box Model, Understanding the Box Model: Content, Padding, Border, Margin, Border Properties: border-width, border-style, border-color, Margin and Padding Properties, Box Sizing: content-box vs border-box; CSS Typography, Font Properties: font-family, font-size, font-weight, font-style, Google Fonts and Custom Fonts, Text Properties: color, line-height, letter-spacing, word-spacing, text-align, text-indent, text-decoration, Units for Fonts: px, em, rem, vh, Responsive Typography; CSS Colors and Backgrounds, Color Formats: Named Colors (RGB, RGBA, HEX, HSL, HSLA), Background Properties: background-color, background-image, background-repeat, background-size, background-position, background-attachment, background-clip.

Module 3 – CSS Advanced Concepts : CSS Layouts, Display Property: block, inline, inline-block, none, Positioning Elements: static, relative, absolute, fixed, sticky, Float and Clear Properties; Flexbox Layout: Flex Container: display: flex, Flex Items: justify-content, align-items, align-self, Flex Direction and Flex Wrap, Ordering Flex Items; CSS Grid Layout: Grid Container: display: grid, Defining Rows and Columns,Grid Gaps and

Alignment, Responsive Grids, Multi-Column Layout; CSS Visual Effects, Opacity and Transparency, Shadows: box-shadow, text-shadow, Gradients:, Linear Gradients, Radial Gradients, Conic Gradients, CSS Filters: blur, brightness, contrast, etc., Masking and Clipping: clip-path, CSS Shapes; CSS Transitions: Adding Smooth Effects with transition-property, transition-duration, and transition-timing-function, CSS Animations: Keyframes: @keyframes, Animation Properties: animation-name, animation-duration, animation-iteration-count, animation-delay, Hover Effects and Interactive UI.

Module 4 - Intermediate Topics : Introduction to Responsive Web Design (RWD), Media Queries: Breakpoints, Mobile-First and Desktop-First Design, Viewport Meta Tag, Responsive Units: vh, vw, em, rem, Responsive Images: max-width and object-fit, Flexbox and Grid for Responsive Layouts, CSS Variables (Custom Properties): Defining and Using Variables: --variable-name, var() Function, CSS Nesting, Modern Layout Techniques: place-items, place-content, gap, Logical Properties for Directional Independence: margin-inline, padding-block, Container Queries (Upcoming Feature): Responsive layouts based on container size, CSS Frameworks and Preprocessors, Overview of CSS Frameworks: Bootstrap, Tailwind CSS, Bulma, CSS Preprocessors: Introduction to Sass and SCSS, Variables, Nesting, Mixins, and Functions, Compiling SCSS to CSS.

Module 5 - Miscellaneous : CSS Best Practices, Writing Clean and Maintainable CSS, BEM (Block Element Modifier) Methodology, Organizing CSS Files for Large Projects, Performance Optimization: Reducing CSS File Size, Avoiding Unused CSS; Project-Based Learning : Personal Portfolio Website, Responsive Landing Page, Interactive Product Card, Fully Responsive Multi-Page Website, Animated and Interactive Dashboard UI, E-commerce Product Listing Page.

SASS

Module 1 - SASS Basics : Overview of SASS, Benefits of using SASS over CSS, Installing SASS (command-line and GUI tools); **SASS Syntax -** SCSS vs. Indented Syntax, Compiling SASS to CSS, Tools for SASS compilation (e.g., Dart SASS, Prepros, Live SASS Compiler for VS Code); **Setting Up a SASS Project -** Directory structure, Best practices for SASS file organization.

Module 2 - SASS Core Concepts : Variables - Defining and using variables, Scope of variables, Use cases for variables in design consistency; Nesting - Nested rules, Avoiding over-nesting pitfalls; Partials and Import - Creating partials, Using @import to organize stylesheets, Limitations of @import and the modern @use and @forward;

Mixins - Creating reusable styles with @mixin, Using @include to call mixins, Arguments and default values in mixins.

Module 3 - SASS Advanced Concepts : Inheritance - Extending styles with @extend, Placeholder selectors, Differences between @extend and mixins; Control Directives - Conditional logic with @if, @else, and @else if, Iteration with @for, @each, and @while, Practical examples of control directives; Functions - Built-in SASS functions (e.g., lighten, darken, percentage, etc.), Creating custom functions; Maps - Working with maps (key-value pairs), Retrieving values from maps, Practical use cases of maps.

Module 4 - SASS Intermediate Concepts : Modular Design with SASS - Structuring styles using the 7-1 pattern (base, components, layout, pages, themes, abstracts, vendors); Error Handling and Debugging - Using @debug and @warn, Common issues and troubleshooting; Responsive Design with SASS - Media query management, Breakpoint mixins and functions; SASS Libraries - Leveraging popular SASS libraries like Bourbon or Compass, Creating and sharing your own SASS library.

Module 5 - Miscellaneous : Performance Optimization - Avoiding unnecessary nesting, Efficient use of variables and mixins; Theming with SASS - Creating and managing multiple themes, Using maps and functions for dynamic theming; SASS with Modern CSS - Combining SASS with modern CSS features (e.g., CSS Grid, Flexbox), Using SASS alongside CSS frameworks like Bootstrap or Materialize; Build Tools and Workflows - Integrating SASS into modern build tools like Webpack, Gulp, or Parcel, Automating SASS compilation and live reloading.

JavaScript

Module 1 - JavaScript Basics: Introduction to JavaScript, History and evolution of JavaScript, Features and use cases, Role of JavaScript in web development (Client-side vs. Server-side), Installing a browser (Chrome/Firefox) and using Developer Tools, Setting up a code editor (Visual Studio Code, Sublime, etc.), Running JavaScript in the browser console, Writing your first script in HTML, Statements and comments, Variables (var, let, const), Data types and type conversion, Primitive data types: string, number, boolean, null, undefined, Checking types using typeof, Operators, Arithmetic, Comparison, Logical, Assignment, Special operators like typeof, instanceof, Conditional statements: if, else if, else, switch, Loops, for, while, do...while, break and continue, Working with numbers in loops (common patterns like sum, multiplication).

Module 2 - Core JavaScript Concepts: Declaring and invoking functions, Parameters and default values, Function expressions and anonymous functions, Function scope and return statement, Immediately Invoked Function Expressions (IIFE), Arrays, Declaring and initializing arrays, Common array methods: push, pop, shift, unshift, splice, slice, concat, join, etc., Iterating over arrays using for and forEach, Multidimensional arrays, Objects, Declaring objects and accessing properties, Adding, updating, and deleting properties, Nested objects and deep access, Iterating over object keys and values using for...in, Strings, String manipulation and methods, length, toUpperCase, toLowerCase, charAt, substring, slice, indexOf, lastIndexOf, split, replace, includes.

Module 3 - Advanced JavaScript Concepts: Introduction to DOM, Accessing DOM elements, Using getElementById, getElementsByClassName, getElementsByTagName, Using querySelector and querySelectorAll, Manipulating DOM elements : Changing content with innerHTML and textContent, Adding/removing classes, Changing styles dynamically, Events and event handling, Common events: click, mouseover, keydown, change, Adding event listeners, Error Handling: try, catch, finally, Throwing custom errors, Understanding common JavaScript errors (e.g., ReferenceError, TypeError), Dates and Times, Working with the Date object, Formatting dates, Calculating time differences, Math, Common Math methods: Math.random, Math.round, Math.floor, Math.ceil, Math.max, Math.min.

Module 4: Intermediate Topics : Introduction to JSON, JSON syntax and parsing, Converting objects to JSON (JSON.stringify) and back to objects (JSON.parse), Storage : localStorage and sessionStorage, Storing, retrieving, and removing data, Use cases for each, Asynchronous JavaScript: Introduction to synchronous vs. asynchronous execution, Using setTimeout and setInterval, Understanding the event loop.

ECMAScript

Module 1 - ECMAScript Basics: Introduction to ECMAScript, History and relationship with JavaScript, Role of ECMAScript in modern JavaScript development, Versions of ECMAScript (ES5 to ESNext), Key milestones and features introduced in major versions; Block-Scoped Declarations: Difference between var, let, and const, Temporal Dead Zone (TDZ), Best practices for using let and const; Arrow Functions: Syntax and differences from regular functions, Implicit returns, Lexical this and common use cases, Limitations of arrow functions (e.g., no arguments object); Template Literals: Multi-line strings, String interpolation using ${}, Tagged template literals (advanced); Destructuring: Array destructuring -Assigning multiple variables, Skipping elements; Object destructuring - Assigning to variables with different names, Default values, Nested destructuring; Default Parameters in Functions: Setting default values for function parameters, Combining default parameters with destructuring; Rest operator

(...) for - Function arguments, Collecting elements into arrays, Object properties; Spread operator (...) for- Merging arrays, Merging objects, Copying arrays and objects.

Module 2 - Core ECMAScript Concepts: Object Enhancements: Shorthand property names, Shorthand method definitions, Computed property names, Object assign and merging objects; Array Enhancements : Array methods: find, findIndex, includes, Copying arrays using spread ([...array]), Filling arrays with Array.fill, Iterators and Iterables : Understanding the iterable protocol, Using the for...of loop, Iterating over strings, arrays, and objects; Generators : Creating and using generator functions, Yielding values with yield, Practical use cases for generators (e.g., lazy evaluation).

Module 3 – Advanced ECMAScript Concepts : Introduction to JavaScript modules, Benefits of modular programming, Key terms: import and export, Using Modules : Default exports vs. named exports, Importing and exporting in ES6, Importing all exports using import * as; Dynamic Imports: Using import() for dynamic module loading, Practical use cases for dynamic imports (e.g., code-splitting); **Promises :** Understanding the Promise constructor, States of a promise: pending, fulfilled, rejected, Chaining promises with .then(), .catch(), and .finally(), Common pitfalls in promises (e.g., unhandled rejections); **Async/Await :** Converting promises to async/await syntax, Error handling with try...catch, Using await with Promise.all and Promise.race.

Module 4 – Intermediate Topics : Introduction to Classes : Class syntax, Defining constructors, Adding methods and properties; Introduction to Inheritance : Using extends for class inheritance, Calling parent class methods using super, Overriding methods in child classes; Static Methods and Properties : Defining static methods and properties, Practical use cases of static methods; New Built-in Objects and Methods - Introduction to Map: Creating and managing key-value pairs, Methods: set, get, has, delete, clear; Introduction to Set: Adding and checking elements, Use cases for unique values; WeakMap and WeakSet, Differences from Map and Set, Use cases for WeakMap and WeakSet; Symbol : Creating and using Symbol as unique identifiers, Practical use cases (e.g., custom object keys).

Module 5 - Miscellaneous : Understanding Proxy, Intercepting and customizing object behavior, Common traps: get, set, deleteProperty; Using Reflect for default object operations; Dynamic Property Access, Object.getOwnPropertyDescriptors, Using Object.entries and Object.values, ESNext Features - Optional Chaining: Using ?. for null-safe property access, Combining with default values (??); Nullish Coalescing, Difference between ?? and || ; Logical Assignment Operators, ||=, &&=, ??=, Promise.allSettled, Use cases for handling multiple promises with allSettled, Introduction to BigInt, Arithmetic with BigInt.

JSON

Module 1 – JSON Basics : Introduction to JSON, History and Evolution of JSON, Why JSON?, Lightweight Data Interchange Format, Comparison with XML, Key Features of JSON, Understanding Syntax Rules; JSON Data Types: Strings, Numbers, Booleans, Arrays, Objects, Null, JSON Structure: Key-Value Pairs, Nesting Objects and Arrays.

Module 2 – JSON Core Concepts : Using JSON in JavaScript, Parsing JSON: JSON.parse(), Stringifying JSON: JSON.stringify(), Handling Errors with try...catch, Accessing JSON Data, Dot Notation vs. Bracket Notation, Iterating JSON Arrays and Objects, for Loop, forEach() and map().

Module 3 – JSON Advanced Concepts : Understanding REST APIs and JSON, Fetching JSON Data from APIs, Using fetch() in JavaScript, Handling Promises with .then(), Error Handling with catch(), Using async/await for Fetching Data, Validating JSON Syntax: Online JSON Validators, JSON Lint Tools; Handling Errors with Invalid JSON, Common Mistakes and Debugging; Storing JSON Data in Local Storage, localStorage.setItem(), Retrieving JSON Data from Local Storage, localStorage.getItem(), Converting Objects/Arrays to JSON for Storage, Deleting JSON Data from Local Storage.

Module 4 – Intermediate Topics : Destructuring Arrays and Objects, Combining JSON Data with Modern ES6 Features, Spread Operator (...), Rest Operator, JSON Schema, Defining and Validating JSON Data, Nested JSON Handling: Accessing and Manipulating Deeply Nested Data, Flattening JSON Structures; JSON Tools, JSON Formatter and Beautifier, JSON Minifier, JSON Editor Tools, Converting JSON to CSV, XML, and Other Formats.

Fetch API & Axios

Module 1 - Introduction to HTTP and APIs : Introduction to API, REST APIs: Definition and examples, Understanding JSON and how APIs communicate; Introduction to HTTP - HTTP methods (GET, POST, PUT, DELETE, etc.), HTTP status codes, Headers, request body, and query parameters, Difference between client-side and server-side APIs.

Module 2 - Fetch API : Introduction to Fetch API, Why use Fetch API over XMLHttpRequest, Syntax of fetch(); Making Requests - Simple GET request, Handling responses : response.json(), response.text(), response.blob(), response.arrayBuffer(), Sending data with POST requests : Setting request headers, Sending JSON in the request body; Handling Errors - Checking HTTP status codes, Handling network errors, Using try...catch for error handling; Fetch Options - Specifying HTTP methods (GET, POST, PUT, etc.), Adding custom headers, Sending cookies with credentials (same-origin, include, omit); Advanced Fetch Features - Working with ReadableStream for large data, Abort requests with AbortController, Implementing request timeouts, Handling redirects with redirect property; Practical Examples : Fetching data from a public API (e.g., JSONPlaceholder, OpenWeatherMap), Fetching paginated data, File download using Fetch API.

Module 3 – Axios : Introduction to Axios, why use it over Fetch API, Installing Axios with npm or CDN, Comparing Axios with Fetch API; Making Requests with Axios - Simple GET request, Sending POST requests, JSON payloads, Form data submission, PUT, PATCH, and DELETE requests, Handling Responses - Working with response objects, Accessing data, status, headers, etc., Error handling with Axios, Using catch for errors, Differentiating between HTTP errors and network errors; Configuring Axios - Setting global defaults : Base URL, Default headers, Timeout settings, Creating Axios instances for API management,Overriding default configurations for specific requests; Interceptors - Using request interceptors, Adding authentication tokens, Modifying request headers dynamically, Using response interceptors, Centralized error handling, Transforming response data; Axios Utility Methods - axios.all and axios.spread for concurrent requests, Canceling requests with CancelToken or AbortController, Handling file uploads and downloads with Axios; Practical Examples - Consuming an API with multiple endpoints, Creating a reusable Axios wrapper, Handling API retries for failed requests, Building a search functionality with debouncing.

Module 4 – Intermediate Topics : Syntax comparison between Fetch and Axios, Error handling differences, Advantages and limitations of each approach, When to choose Fetch API vs. Axios; API Integration Techniques - Common Patterns, CRUD operations with APIs, Pagination and infinite scrolling, Filtering and searching API data; Security Best Practices - Protecting sensitive information (e.g., API keys), Using https and CORS, Avoiding Cross-Site Scripting (XSS) and other vulnerabilities, Performance Optimization : Minimizing API calls with caching, Debouncing and throttling API requests, Optimizing large data transfers with gzip and chunking; Error Handling and Debugging - Debugging API calls with browser developer tools, Logging API errors, Implementing retry logic with exponential backoff, Showing user-friendly error messages; Testing API Integrations - Manual testing with tools like Postman or Insomnia, Writing unit tests for Fetch and Axios using Jest, Mocking API calls in tests.

Git & GitHub

Module 1 - Version Control & Git Basics : Introduction to Version Control, Importance of Version Control in Software Development, Types of Version Control Systems: Centralized VCS (e.g., SVN), Distributed VCS (e.g., Git), Overview of Git and GitHub; Installing Git: Setting up Git on Windows, macOS, and Linux; Configuring Git for the First Time: git config --global user.name, git config --global user.email; Git Core Concepts: Repository, Commit, Branch, Working Directory, Staging Area, and Commit History, Creating a Local Repository (git init), Adding and Committing Changes (git add, git commit), Viewing Commit History (git log, Filtering Logs (--oneline, --graph, --author)), Checking Repository Status (git status).

Module 2 – Git Core Concepts : Working with Git Files, Tracking New, Modified, and Deleted Files, Ignoring Files (.gitignore File and Patterns), Viewing Changes (git diff, Comparing Staged and Committed Changes), Undoing Changes (Resetting Files: git reset, git restore, Reverting Commits: git revert, Undoing Last Commit), Git Branches, Creating and Switching Branches (git branch, git checkout, git switch), Merging Branches (Fast-Forward vs. 3-Way Merge, git merge), Handling Merge Conflicts (Identifying and Resolving Conflicts), Deleting Branches (Local Branch: git branch -d, Remote Branch: git push origin –delete).

Module 3 – Git Advanced Concepts : Remote Repositories (GitHub), Setting Up a GitHub Account, Connecting Local Git with GitHub (Creating a Remote Repository, git remote add origin), Pushing Code to GitHub (git push), Cloning Repositories (git clone), Pulling Changes from Remote (git pull), Forking and Starring Repositories, Exploring GitHub UI (Repositories, Issues, Pull Requests, Actions, and Insights), Collaborating with GitHub, Fetching and Merging Changes (git fetch and git pull), Creating Pull Requests (PRs) (Forking, Cloning, and Opening a Pull Request, Reviewing and Merging PRs), Code Reviews and Discussions, Resolving PR Merge Conflicts, Understanding GitHub Labels, Milestones, and Assignees.

Module 4 – Intermediate Topics : Understanding Tags in Git (Lightweight Tags, Annotated Tags), Creating and Managing Tags (git tag, git tag -a), Pushing Tags to Remote Repository, Creating Releases on GitHub, GitHub Issues (Creating, Assigning, and Managing Issues), Advanced Git Commands : Rebasing Branches (git rebase vs. git merge), Cherry-Picking Commits (git cherry-pick), Stashing Changes (git stash and Managing Stashes), Amending Commits (git commit –amend), Squashing Commits (Merging Multiple Commits into One).

Module 5 - Miscellaneous : Security in Git and GitHub, Managing Sensitive Information: Avoiding Secrets in Code, Using .gitignore Properly; Encrypting Git Commits, Signing Commits with GPG, Protecting Branches: Branch Protection Rules in

GitHub; Latest GitHub Features: GitHub Copilot for AI-Powered Coding, GitHub Codespaces for Cloud Development, GitHub Actions Enhancements, New Git Commands and Improvements: git restore and git switch (Modern Alternatives), Improved Handling of Large Files with Git LFS.

React Js

Module 1 – React Basics : Introduction to React, The history and evolution of React, Key features of React (Declarative, Component-Based, Learn Once Write Anywhere), Overview of React ecosystem: React Router, Redux, React Query, etc.; Understanding React's Virtual DOM - What is Virtual DOM, Difference between Virtual DOM and Real DOM, How React updates the DOM efficiently; What's New in React 18 - Concurrent rendering, Automatic batching, Transitions and startTransition; Setting Up the Development Environment - Installing Node.js and npm, Creating a new React application: create-react-app, Using Vite for modern React projects, Understanding project structure, Setting up VS Code with React extensions, Installing and using ESLint and Prettier.

Module 2 – React Core Concepts : What is JSX, Writing JSX: Basic syntax and embedding JavaScript, Expressions and conditional rendering in JSX, JSX rules and common pitfalls, Parent wrapping element, Reserved words in JSX (className, htmlFor), Understanding React.createElement under the hood; React Components - Functional Components : What are components, Creating and rendering functional components, Best practices for structuring components, Passing and receiving props, Prop validation with PropTypes, Default props, Component composition and reusability, Prop Drilling; Class Components (Legacy Support) : Understanding class components for legacy systems, Lifecycle methods overview : componentDidMount, componentDidUpdate, componentWillUnmount, Comparison between class components and functional components. Styling React Applications - Inline styling and CSS modules, Using styled-components for CSS-in-JS, Integrating CSS frameworks like Tailwind CSS, Responsive design principles with React; Animation in React - Adding basic animations with CSS, Using React animation libraries: React Transition Group, Framer Motion; Error Handling - Error boundaries in React, Catching JavaScript errors, Creating reusable error boundaries, Handling errors in API calls and promises; absolute paths & relative paths, Differences between absolute and relative paths in React, Importing a component using an absolute path, Importing a component using a relative path, Advantages and disadvantages of each approach.

Module 3 – React Advanced Concepts : React State Management - What is state, Managing local state with useState, Using arrays and objects in state, Immutability in state updates, Conditional rendering based on state, Debugging state with React Developer Tools, Best practices for organizing state in components; React Hooks - Basic Hooks : useState (Managing local state, Working with primitive, array, and object states), useEffect (Managing side effects, Data fetching, Subscriptions and cleanup, Dependency arrays and their importance), useContext (Context API with hooks, Avoiding prop drilling using useContext), Additional Hooks : useReducer (complex state management, Comparison with useState, Creating a reducer function, Lazy initialization), useRef (Accessing DOM elements and persisting values across renders, Managing focus, timers, and animations), useMemo (Memoizing expensive computations), useCallback (Optimizing callback functions, Comparison with useMemo), Advanced Hooks – useLayoutEffect (Implemetation, Difference from useEffect, Use cases in synchronizing with DOM changes), useImperativeHandle (Customizing instance values, Usage with React.forwardRef), useId (Generating unique IDs for accessibility and forms), useTransition (Managing concurrent UI updates, Using transitions for smooth UI state changes), useDeferredValue (Defer updates for better performance), useSyncExternalStore (Reading external store states), useInsertionEffect (Optimizing CSS-in-JS libraries).

Module 4 – Intermediate Topics : What is Context API, Avoiding prop drilling, Creating and providing context, Consuming context using useContext, Structuring context for scalable applications, Combining Context API with reducers (useReducer); React Lifecycle - Understanding React's rendering cycle, React lifecycle in functional components, Mounting, updating, and unmounting stages, Debugging component lifecycle with React DevTools; Event Handling - Handling DOM events in React: onClick, onChange, onSubmit, Passing arguments to event handlers, Synthetic events vs. native events, Preventing default behavior and stopping event propagation; Introduction to routing in React, Setting up react-router-dom, Defining routes with Routes and Route, Navigating programmatically with useNavigate, Nested routes and layout components, Dynamic routing with URL parameters, Protected routes for authentication, API Integration - Fetching data with useEffect, Using third-party libraries like Axios, Handling loading, success, and error states, Optimizing API calls with React Query, Query caching, Paginated data fetching; React Performance Optimization - Optimizing re-renders with React.memo, Using useMemo and useCallback effectively, Lazy loading components with React.lazy and Suspense, Using Profiler for performance debugging, React's concurrent rendering and automatic batching.

Module 5 - Miscellaneous : Building Reusable Components - Best practices for creating reusable and modular components, Component prop patterns: Controlled and uncontrolled components, Compound components, Higher-order components (HOCs), Custom hooks for reusable logic; Testing React Applications - Setting up testing libraries: Jest, React Testing Library, Writing unit tests for components, Testing hooks, Snapshot testing for UI consistency; Deployment : Optimizing React apps for production: Code splitting, Tree shaking; Hosting on platforms like Netlify, Vercel, or AWS, Configuring CI/CD pipelines for React apps.

React Query (TanStack)

Module 1 - React Query Basics : Overview and Key Features, Benefits of Using React Query in Applications, Comparison with Other Data Fetching Libraries (Axios, Fetch); Setting Up React Query - Installing React Query, Setting Up QueryClient and QueryClientProvider, Basic Example: Fetching Data with useQuery.

Module 2 - React Query Core Concepts : Query Basics - Understanding useQuery, Configuring Queries (Keys, Functions), Query Status (isLoading, isError, isSuccess); Mutations - What is a Mutation?, Using useMutation for Data Updates, Handling Optimistic Updates, Error Handling with Mutations; Query Invalidation - Understanding Query Caching, Manually Invalidating Queries, Re-fetching Queries on Demand; Query Keys - Importance of Query Keys, Structuring Complex Query Keys, Best Practices for Unique Key Generation.

Module 3 - React Query Advanced Concepts : Caching and Stale Data - Understanding Cache Behavior, Configuring Cache Time and Stale Time, Cache Invalidation on Data Change; Pagination and Infinite Queries - Implementing Pagination with useQuery, Using useInfiniteQuery for Infinite Scrolling, Managing Cursors and Offset-Based Pagination; Dependent Queries - Chaining Queries with Dependencies, Conditional Fetching with enabled Option; Prefetching Data - Using queryClient.prefetchQuery, Prefetching for Smooth Navigation; Background Fetching - Query Refetch Intervals, Displaying Background Loading States (isFetching); Data Transformation - Transforming Data with Selectors (select Option), Memoizing Query Results.

Module 4 - React Query Intermediate Concepts : Handling Large Datasets - Efficient Data Handling Strategies, Server-Side Pagination vs Client-Side Filtering; Avoiding Over-fetching - Using Query Keys for Selective Fetching, Configuring Query Behavior (Retries, Timeout); Optimistic Updates - How to Implement Optimistic Updates for Mutations, Reverting Changes on Errors; React Query DevTools - Installing and Using DevTools, Debugging Query States and Cache.

Module 5 – Miscellaneous : Error Handling - Handling Errors Globally, Custom Error Boundaries for Queries; Persisting Query Data - Using react-query-persist-client, Persisting Cache Between Page Loads; SSR and React Query - Setting Up React Query for Server-Side Rendering, Hydration of Query Data on the Client; Authentication and Protected Queries - Fetching Authenticated Data, Handling Tokens and Expired Sessions; Testing React Query - Testing Components with Queries and Mutations, Mocking QueryClient and API Responses; Integration with Other Libraries - Using React Query with Axios or Fetch, Integrating with GraphQL APIs.

React Testing Library

Module 1 - React Testing Library (RTL) Basics : Introduction to Testing in React, Why React Testing Library?, Core principles of RTL: testing behavior vs. implementation.; Setup and Installation - Installing Jest and React Testing Library, Setting up a React project for testing (with create-react-app or manual configuration), Adding @testing-library/jest-dom for extended assertions; Understanding Basic RTL Functions - Rendering components with render, Cleaning up rendered components automatically, The role of screen for finding elements; Basic Test Case Writing - Writing your first test case with RTL, Introduction to Jest matchers (toBe, toHaveTextContent, etc.), Structuring test files and organizing test cases; Key Concepts of Queries - Types of queries: Role-based: getByRole, Text-based: getByText, getByPlaceholderText, Label-based: getByLabelText, Differences: getBy, queryBy, and findBy, Best practices for choosing queries.

Module 2 - RTL Core Concepts : Simulating User Interactions - Using fireEvent to simulate DOM events (click, change, submit), Using userEvent for more realistic interactions (type, select, drag-and-drop); Testing Forms - Testing input fields: text, checkbox, radio buttons, and dropdowns, Testing controlled vs. uncontrolled components, Simulating form submissions and validations; Asynchronous Code Testing - Handling asynchronous actions with waitFor, Using findBy for elements rendered asynchronously, Mocking async functions like fetch and axios; Snapshot Testing - Creating snapshots with Jest and RTL, Updating and managing snapshots, When to use (and avoid) snapshot tests; Testing Error States - Mocking errors in API responses, Testing components with error boundaries, Verifying fallback UI in error scenarios.

Module 3 - RTL Advanced Concepts : Mocking External Dependencies - Using Jest to mock modules and functions, Mocking API calls with jest.mock() and msw, Mocking third-party libraries (like Redux, React Query); Testing Context and Providers -

Understanding React Context in tests, Overriding Context values in tests, Testing nested components using Providers; Testing React Hooks - Writing tests for custom hooks, Mocking built-in hooks like useState, useEffect, and useContext, Testing hooks that depend on external APIs; Component Composition Testing - Testing parent-child relationships, Mocking child components for isolated testing, Verifying props passed to children; Testing Routing - Setting up React Router for testing, Simulating navigation with MemoryRouter, Verifying route changes and dynamic parameters.

Module 4 – RTL Intermediate Concepts : Accessibility Testing with RTL - Verifying ARIA roles and attributes, Using getByRole for accessible queries, Identifying and fixing accessibility issues; Testing Complex UI Interactions - Testing modals, tooltips, and dropdowns, Simulating drag-and-drop functionality, Verifying animations and transitions; Integration Testing - Combining multiple components in tests, Mocking external APIs in integration tests, Testing component interaction and data flow; Global State Management Testing - Testing components using Redux or Context API, Mocking store and dispatch actions, Verifying state changes in connected components; Testing Performance and Optimization - Identifying performance bottlenecks in tests, Using Jest timers for testing debounce/throttle functions, Testing React.memo and useMemo optimizations.

Module 5 – Miscellaneous : Structuring Tests for Maintainability - Organizing test files and directories, Writing reusable utility functions for tests, Using setup functions for repetitive tasks; Debugging Tests - Using screen.debug() to inspect rendered components, Common errors in RTL and how to resolve them, Tips for debugging flaky tests; Code Coverage and Reporting - Generating code coverage reports with Jest, Interpreting coverage metrics (statements, branches, functions), Improving coverage without over-testing; Integrating with CI/CD Pipelines - Setting up tests to run in CI environments (e.g., GitHub Actions, Jenkins), Optimizing test runtime in pipelines, Reporting and analyzing test results in CI.

TypeScript

Module 1 - TypeScript Basics : Introduction to TypeScript, Difference between TypeScript and JavaScript, Benefits of using TypeScript, Installing TypeScript, Using npm or yarn, Setting up a TypeScript project, TypeScript Playground: An interactive way to test TypeScript code, Understanding Types : What are types, and why are they important, Primitive types - string, number, boolean, null, undefined; Special types - any, unknown, never, void; Type Annotations : Explicit type annotations for variables, Type inference: How TypeScript infers types automatically; Union and Intersection

Types : Union types (|), Intersection types (&), Practical examples of unions and intersections; Literal Types : String, number, and boolean literals, Combining literals with union types.

Module 2 – TypeScript Core Concepts : Typed arrays, Tuples - Fixed-length arrays with specific types, Optional tuple elements; Enums : Numeric and string enums, Using enums in real-world scenarios, Enum vs. literal types; Type Aliases : Creating reusable custom types, Combining type aliases with union and intersection types; Interfaces : Defining interfaces, Optional and readonly properties, Extending interfaces, Difference between interfaces and type aliases; Function Types : Typing function parameters and return values, Optional and default parameters, Function overloading; Classes : Defining classes with TypeScript, Constructor typing, Access modifiers: public, private, protected, readonly; Inheritance : Extending classes, Method overriding, Using super for parent class methods, Abstract Classes : Defining abstract classes and methods, Implementing abstract classes; Interfaces with Classes : Implementing interfaces in classes, Using multiple interfaces in a class.

Module 3 – TypeScript Advanced Concepts : Introduction to Generics, why use Generics, Generic functions and constraints, Generic types in arrays and objects, Defining generic interfaces, Generic classes with real-world examples; Utility Types : Partial, Required, Readonly, Pick, Omit, Using utility types to manipulate object structures; TypeScript Utility and Advanced Features - Type Assertions : Using as for type assertions, Difference between type assertions and type casting; Index Signatures : Defining objects with dynamic keys, Typing objects with unknown keys; Mapped Types : Creating new types based on existing ones, Examples with keyof, in, and indexed access; Conditional Types : Syntax and use cases, Using infer to extract types, Declaration Merging : Combining multiple interface declarations, Practical scenarios for declaration merging; Modules : Importing and exporting in TypeScript, Namespaced modules vs. ES6 modules, Configuring module resolution in tsconfig.json; Defining and using namespaces, Merging namespaces with classes and interfaces.

Module 4 – Intermediate Topics : TypeScript Configuration and Tools - tsconfig.json, Key properties in tsconfig.json: target, module, strict, paths, baseUrl, Setting up a strict type-checking environment; TypeScript Compiler : Compiling TypeScript to JavaScript, Using the tsc command-line tool, Watching files for changes with tsc –watch; Error Handling and Debugging - Type Errors : Understanding compile-time errors, Fixing common type-related issues, Debugging TypeScript : Using source maps for debugging, Debugging TypeScript in modern browsers and editors; Working with External Libraries - Using Type Definitions : Installing type definitions from @types, Managing external libraries with npm and yarn; Writing Custom Type Definitions :

Creating .d.ts files for third-party libraries, Exporting types for use in TypeScript projects.

Module 5 - Miscellaneous : Mixins: Creating and using mixins in TypeScript, Combining multiple classes with mixins; Decorators : Introduction to decorators, Using class, method, and property decorators, Practical examples (e.g., logging, validation), TypeScript with Front-End Frameworks - TypeScript with React (Optional) : Typing React components, Using hooks with TypeScript; TypeScript with Angular (Optional) : TypeScript as a core language in Angular, Typing services and components.

Next Js

Module 1 – Next.js Basics : Introduction to Next.js, Why use Next.js over React, Key features and advantages of Next.js, Difference between client-side rendering (CSR), server-side rendering (SSR), and static site generation (SSG), Overview of the Next.js ecosystem: File-based routing, API routes, Middleware and Edge Functions; Setting Up a Next.js Project - Installing Node.js and npm, Creating a Next.js application using: npx create-next-app, Template projects (with TypeScript and ESLint), Understanding the Next.js project structure, Setting up development tools: VS Code extensions for Next.js, ESLint and Prettier for code formatting.

Module 2 – Next.js Core Concepts : Introduction to file-based routing, Creating pages in the pages directory, Dynamic routes: Catch-all routes ([...params]), Optional catch-all routes ([[...params]]), Nested routes and layout pages, Using the Link component for navigation, Customizing the 404 and 500 error pages; Rendering Methods in Next.js - Server-Side Rendering (SSR) : What is SSR, Using getServerSideProps for server-side data fetching, Benefits and trade-offs of SSR, Static Site Generation (SSG) : What is SSG, Using getStaticProps for pre-rendering pages, Dynamic routes with getStaticPaths, Incremental Static Regeneration (ISR): Updating static content at runtime, Configuring revalidation time; Client-Side Rendering (CSR) : When to use CSR in a Next.js application, Fetching data with hooks like useEffect and useSWR; Data Fetching in Next.js - Overview of Next.js data-fetching methods: getServerSideProps, getStaticProps, and getStaticPaths, Fetching data using REST APIs and GraphQL, Using useSWR for client-side data fetching: Handling caching, revalidation, and error states, Streaming responses with react-server-components (RSC).

Module 3 – React Advanced Concepts : Layouts in Next.js - Creating shared layouts for multiple pages, Using getLayout for per-page layouts, Nested layouts with the app directory structure; Head Management - Adding metadata using the Head component, Managing SEO tags dynamically, Open Graph and Twitter card meta tags; Styling in Next.js - Styling options in Next.js: Global CSS, CSS Modules, Styled-Components (CSS-in-JS), Integrating Tailwind CSS for utility-first styling, Optimizing styles for critical rendering paths; Next.js Middleware : What is middleware in Next.js, Writing middleware functions: Request and response lifecycle, Examples: Redirecting users, Securing routes with authentication; Deploying middleware at the Edge; API Routes in Next.js - Creating API endpoints in the pages/api directory, Handling HTTP methods (GET, POST, etc.), Parsing request body and query parameters, Middleware integration in API routes, Authentication, CORS handling, Connecting to a database (e.g., MySQL, MongoDB, PostgreSQL); Next.js and Authentication - Overview of authentication approaches: Session-based vs. token-based authentication, Implementing authentication using NextAuth.js, Securing API routes and pages, Role-based access control in Next.js.

Module 4 – Intermediate Topics : Image Optimization - Using the next/image component: Automatic resizing and lazy loading, Supporting multiple image formats (e.g., WebP), Configuring external image loaders, Best practices for performance optimization; Static Assets and Fonts - Managing static files in the public folder, Integrating custom and Google Fonts using the next/font library, Optimizing font loading for performance; Internationalization (i18n) - Built-in internationalization support in Next.js, Configuring locales in next.config.js, Routing based on locale, Using translation libraries (e.g., react-i18next); Performance Optimization in Next.js - Measuring performance with Lighthouse and Web Vitals, Optimizing large Next.js applications: Code splitting and dynamic imports (next/dynamic), Tree-shaking, Caching strategies, Pre-rendering and lazy loading techniques; Edge Functions - What are Edge Functions, Deploying serverless functions to the Edge, Examples: Geo-based personalization, A/B testing; React Server Components (RSC) - Introduction to React Server Components in Next.js, Differences between client and server components, Combining RSC with SSR and SSG, Streaming - Streaming server-rendered pages with Suspense and React.lazy, Improving performance with partial rendering.

Module 5 - Miscellaneous : Testing in Next.js - Writing unit tests for components and pages using Jest, Testing data-fetching methods (getServerSideProps, getStaticProps), End-to-end testing with Cypress or Playwright, Debugging tests with mocked APIs; Deploying Next.js Applications - Preparing the app for production: Building the app (next build), Environment variable management, Deploying on platforms: Vercel (native support), Netlify, or AWS Amplify, Setting up CI/CD pipelines for automated deployments.

Vue Js

Module 1 – Vue.js Basics : Introduction to Vue.js, History and evolution of Vue.js, Advantages of Vue.js over other frameworks, Understanding Vue's ecosystem (Vue CLI, Vue Router, Vuex/Pinia), Setting up the development environment, Installing Node.js and npm, Installing Vue CLI or Vite, Creating and running the first Vue application; Vue Instance - The concept of the Vue instance, Creating and mounting a Vue instance, Vue lifecycle hooks, Commonly used hooks (created, mounted, updated, destroyed); Template Syntax - Understanding Vue templates, Data binding with {{ }}, Using JavaScript expressions in templates, Template directives: v-bind, v-if, v-else, v-else-if, v-for, v-show, v-on; Data and Methods - Declaring reactive data with data(), Using methods for event handling, Difference between reactive and non-reactive properties.

Module 2 – Vue Core Concepts : Introduction to Components, Creating and registering components, Local vs. global components, Nesting components; Props - Passing data to child components using props, Prop validation, Default values for props; Event Emitting - Using $emit to send data/events from child to parent, Custom event handling in parent components; Slots - Using slots for content distribution, Named and scoped slots, Default slot content; Directives and Event Handling - Built-in Directives : v-html, v-cloak, v-pre, v-once, v-model for two-way data binding; Event Modifiers - Using .stop, .prevent, .capture, .self, .once; Key Modifiers - Handling keyboard events with .enter, .tab, .space, etc., Custom key modifiers.

Module 3 – Vue Advanced Concepts : Reusability and Composition – Mixins : Creating reusable functionality using mixins, Global vs. local mixins, Limitations of mixins; Custom Directives - Creating and using custom directives, Lifecycle hooks in custom directives (bind, inserted, update, unbind); Filters - Transforming output in templates, Global and local filters, Deprecation of filters in Vue 3 (alternative approaches); Composition API - Understanding the difference between Options API and Composition API, Using setup function, Reactive state with ref and reactive, Computed properties and watchers in Composition API, Using lifecycle hooks with Composition API; Vue Router - Setting up Vue Router : Installing Vue Router, Creating routes and navigating between pages, Using <router-view> and <router-link>, Advanced Routing : Dynamic routes with route params, Nested routes, Programmatic navigation, Route guards (beforeEach, afterEach); Lazy Loading and Code Splitting - Defining lazy-loaded routes for performance optimization, Preloading and prefetching routes.

Module 4 – Intermediate Topics : Basics of State Management - Understanding the concept of state, Sharing state between components with props and events, Limitations of prop drilling and event propagation; Pinia (Recommended for Vue 3) - Installing and setting up Pinia, Creating and using stores, State, getters, and actions in

Pinia; Vuex (For Vue 2 or legacy projects) - Core concepts: State, getters, mutations, and actions, Modular state management in Vuex, Differences between Vuex and Pinia; Working with APIs - Fetching Data, Using axios or Fetch API with Vue.js, Displaying API data in components, Handling loading states and errors; Advanced API Interactions - Sending data to APIs (POST, PUT, DELETE), Canceling requests with AbortController, Using interceptors with axios; Forms and User Input - Handling Form Inputs, Two-way binding with v-model, Binding inputs, checkboxes, radio buttons, and selects; Form Validation - Basic validation with Vue, Using third-party libraries like VeeValidate or Vue Formulate; Animation and Transitions - Basics of Vue transitions, Enter and leave transitions with transition component, Animating elements with transition-group, Using third-party animation libraries (e.g., GSAP).

Module 5 - Miscellaneous : Testing in Vue.js - Unit Testing : Setting up testing with Jest or Mocha, Writing tests for components, Mocking props and events; End-to-End Testing : Using Cypress for E2E testing, Writing and running E2E tests for Vue apps; Teleport : Rendering components outside of their parent DOM; Vue Plugins : Creating and using Vue plugins, Examples of popular Vue plugins (e.g., Vue Toastification); Error Handling : Handling errors globally with errorHandler, Error boundaries in Vue 3; Performance Optimization : Lazy loading components, Using keep-alive for caching components, Optimizing large applications with Vue DevTools; Building and Deployment : Configuring build tools (vite or Webpack), Environment variables in Vue apps, Deploying Vue applications.

Angular

Module 1 – Angular Basics : Introduction to Angular, Evolution and Ecosystem, Deep dive into Angular's architectural philosophy, Comparison with other frameworks (React, Vue), How Angular handles enterprise-grade applications; Setup and Configuration - Advanced Angular CLI options, Customizing project configurations (angular.json, tsconfig.json), Setting up multi-environment configurations; Advanced TypeScript for Angular - Advanced generics for Angular services and components, Utility types for cleaner interfaces and models, Using keyof, typeof, and mapped types in Angular projects, Type inference in RxJS, Advanced decorators: How Angular uses TypeScript for metadata.

Module 2 – Angular Core Concepts : Advanced module structuring for large applications : CoreModule, SharedModule, and FeatureModules, Creating library modules for reusable components, Optimizing application startup with lazy-loaded modules, Using providedIn and modular services; Component Architecture - Component Design Patterns : Smart (container) and dumb (presentational) components, Breaking down complex components into reusable parts, Dynamic component loading with ComponentFactoryResolver and Angular 16's new APIs; Lifecycle Hooks in Depth : Advanced use of ngOnChanges, AfterContentInit, and AfterViewInit, Performance tuning with lifecycle hooks; Dynamic Components - Injecting components dynamically, Use cases: Modals, notifications, and dashboards; Template Optimization - Using Angular structural directives for better performance, Optimizing templates with Angular's view encapsulation strategies; Custom Directives - Advanced use cases for attribute and structural directives, Interacting with other directives and components.

Module 3 – Angular Advanced Concepts : State Management in Angular, Change Detection - How Angular's change detection works under the hood, Optimizing change detection with OnPush strategy, Using NgZone and zone.js to manage Angular's rendering pipeline; State Management Libraries - Using @ngrx/store for advanced state management, Comparing @ngrx, Akita, and MobX for Angular; Introduction to signals and reactivity, Replacing services with signals for state management; Advanced RxJS Patterns - Higher-order observables, Error handling and retry strategies, Combining observables with merge, concat, switchMap, forkJoin; Custom Observables - Building custom operators, Observable and Subject patterns: BehaviorSubject, ReplaySubject, AsyncSubject; Asynchronous Challenges - Managing race conditions and complex event flows; Routing and Navigation - Advanced Routing Techniques : Preloading strategies for better UX, Router lifecycle events for analytics, Nested and child routing patterns; Dynamic Routing : Dynamically generating routes based on API data, Dynamic modules with Angular's router; Authentication and Authorization : Implementing secure routing with CanActivate, CanLoad, and CanDeactivate, Role-based and permission-based navigation; Complex Forms with Validation - Custom validators and async validators, Cross-field validation patterns; Dynamic Forms - Generating forms from metadata or configuration objects, Reusable dynamic form components; Performance Optimization - Handling large forms with performance considerations, Form state management for enterprise applications; Angular Animations - Deep dive into the @angular/animations library, Advanced animation techniques: Animation triggers, states, and transitions, Sequential and parallel animations, Using animations for route transitions.

Module 4 – Intermediate Topics : Testing in Angular - Unit Testing : Writing comprehensive tests for services, components, and directives, Mocking dependencies with Jasmine and Karma; Integration Testing : Testing complex interactions with dependency injection, Writing tests for dynamic components and forms; End-to-End Testing : Advanced e2e testing with Cypress or Playwright, Mocking APIs for e2e scenarios; Performance Optimization - Advanced Optimization Techniques : Ivy runtime optimizations, Custom preloading strategies, Fine-tuning lazy loading; Reducing Bundle Size : Tree-shaking and code splitting, Removing unused dependencies with webpack-bundle-analyzer; Runtime Performance : Debugging and profiling with Angular DevTools; Advanced Angular Security - Best Practices : Preventing XSS and CSRF attacks, Securing Angular apps with HttpInterceptor; JWT Authentication : Integrating secure token storage, Role-based access control, Advanced Guards : Using guards to enforce security policies dynamically.

Module 5 - Miscellaneous : Progressive Web Apps (PWAs) - Building PWAs with Angular : Adding service workers for offline functionality, Caching strategies with @angular/service-worker; Push Notifications : Sending real-time notifications to Angular apps; Deploying PWAs : Deploying to Firebase or custom servers; Internationalization (i18n) : Advanced i18n with Angular's built-in tools, Dynamic translation loading, Handling locale-specific routing; Advanced Libraries and Tools - Angular Material : Customizing themes and advanced component usage, Using the CDK for drag-and-drop and overlays; Integrating Third-Party Libraries : Integrating charting libraries (e.g., D3.js, Chart.js), Using map APIs (e.g., Google Maps, Leaflet); Building Enterprise Applications - Microfrontend Architecture : Using Angular with microfrontend frameworks like Module Federation, Large-Scale Project Structuring : Managing monorepos with Nx, Handling shared modules and libraries; Versioning and Deployment - Best practices for CI/CD with Angular applications.

Redux, Redux Toolkit & RTK Query

Module 1 - Basics : Introduction to Redux, Why do we need it, Core principles: Single source of truth, state immutability, predictable state, Redux architecture: Store, Actions, Reducers, Dispatch, and Selectors, Differences between Redux and other state management tools (e.g., MobX, Zustand, Vuex); Redux Toolkit (RTK) Basics, Why use Redux Toolkit, Simplified Redux setup with createSlice and configureStore, Writing reducers and actions with createSlice, Using createAsyncThunk for handling asynchronous logic; Introduction to RTK Query, Fetching, caching, and updating data with RTK Query, Writing custom endpoints and queries, Comparing RTK Query with traditional API handling methods (like axios or fetch).

Module 2 - Redux, Redux Toolkit, and RTK Query with React : Setting Up Redux with React - Installing Redux and RTK in a React project, Connecting Redux to React using Provider and useSelector, useDispatch, Structuring a Redux-based React application; Writing slices for modular state management, Managing local and global state with Redux, Optimizing performance: React's memo and Redux's selectors (createSelector); Advanced Redux Patterns in React - Middleware for logging, analytics, or handling side effects, Implementing feature toggles with Redux, Handling forms and complex nested states; RTK Query with React - Integrating RTK Query into React apps, Fetching data with hooks like useQuery and useMutation, Managing query cache, invalidating queries, and polling, Dynamic query arguments and lazy queries, Testing Redux in React - Testing reducers, actions, and async thunks, Testing RTK Query API slices, Writing integration tests for Redux with React Testing Library.

Module 3 - Redux, Redux Toolkit, and RTK Query with Next.js : Setting Up Redux with Next.js, Integrating Redux with the Next.js app architecture, Server-Side Rendering (SSR) with Redux in Next.js, Using getServerSideProps and getStaticProps with Redux; Advanced State Management in Next.js - Managing authentication state with Redux in Next.js, Handling pagination and infinite scrolling with RTK Query, Persisting Redux state across routes; RTK Query in Next.js - Using RTK Query to fetch server-side data, Query caching and invalidation in Next.js applications, Optimizing SSR with pre-fetched data via RTK Query.

Module 4 - Redux, Redux Toolkit, and RTK Query with Vue : Setting Up Redux with Vue, Installing Redux and Redux Toolkit in Vue projects, Integrating Redux with Vue components using vue-redux; Using Redux Toolkit with Vue - Writing slices for modular state management in Vue, Managing Vue component state with Redux; Advanced Redux in Vue - Handling Vue-specific challenges with Redux, Combining Vuex and Redux in hybrid applications, Middleware in Redux for Vue: Logging and analytics; RTK Query with Vue - Fetching API data with RTK Query in Vue, Using query hooks and caching strategies, Optimizing Vue's reactivity with Redux and RTK Query.

Module 5 - Redux, Redux Toolkit, and RTK Query with Angular : Setting Up Redux with Angular, Installing Redux and RTK in an Angular project, Integrating Redux with Angular using @angular-redux; Using Redux Toolkit in Angular - Managing Angular component state with Redux, Writing reducers and actions for Angular-specific use cases; Advanced Redux Patterns in Angular - Handling dependency injection with Redux, Optimizing Angular's performance with OnPush and Redux selectors; RTK Query in Angular - Fetching data in Angular using RTK Query, Querying and caching strategies for Angular apps, Dynamic query arguments and mutation handling.

Jest

Module 1 – Jest Basics : Introduction to Jest, Key Features and Benefits of Jest, Comparison with Other Testing Frameworks (Mocha, Jasmine, etc.), Use Cases of Jest in Front-End Development; Setting Up Jest - Installing Jest in React, Vue, Angular, and Next.js Applications, Creating and Configuring jest.config.js, Setting Up Test Environments (jsdom, Node), Setting Up Module Aliases and Path Mapping; Running Jest Tests from the CLI, Key Commands: --watch, --coverage, --runInBand; Writing Basic Tests - Understanding Test Suites and Test Cases : describe, it, and test, Common Matchers in Jest, Basic Assertions: toBe, toEqual, toMatch, toHaveLength, Truthiness Assertions: toBeTruthy, toBeFalsy, Number and String Matchers; Testing Asynchronous Code - Using done for Callbacks, Testing Promises and Async/Await; Jest CLI Basics - Running Tests with Specific Patterns, --testNamePattern and –testPathPattern, Watching Tests in Development Mode, Debugging Tests with CLI Options (--verbose, --detectOpenHandles).

Module 2 - Jest Core Concepts : Mocking in Jest - What is Mocking?, Creating Mock Functions with jest.fn(), Auto-Mocking Modules with jest.mock(), Manual Mocks for Custom Implementations, Spying on Functions with jest.spyOn, Mocking Axios/Fetch Requests; Testing Front-End Components - Testing React Components, Using @testing-library/react for DOM Interactions, Mocking Props, Context, and Hooks; Testing Vue Components - Using Vue Test Utils with Jest, Testing Slots and Emitters; Testing Angular Components - Configuring Jest with Angular, Mocking Services and Dependencies; Testing Next.js Features - SSR/SSG Testing, API Route Testing with Mock Servers; Event and State Testing; Simulating User Events - fireEvent and userEvent, Testing Button Clicks, Form Inputs, and Checkboxes; Testing State and Props - Testing Prop Updates in Child Components, Mocking Redux/Vuex State, Testing Custom Hooks in React; Snapshot Testing, Creating and Updating Snapshots, Managing Large Snapshots, Using toMatchSnapshot and toMatchInlineSnapshot, Best Practices for Snapshot Tests.

Module 3 - Jest Advanced Concepts : Handling Timers, Using Jest Fake Timers - jest.useFakeTimers vs jest.runAllTimers, Testing Code with setTimeout and setInterval; Testing Debounced and Throttled Functions; Managing API Calls, Mocking REST APIs with Axios/Fetch - Using Mock Responses, Simulating Network Errors; Handling GraphQL APIs- Mocking GraphQL Queries with Apollo Mock Provider; Testing Components with Loading and Error States; Testing Forms, Simulating Form Submissions, Validating Form Inputs - Testing Required Fields, Testing Error Messages, Testing Custom Validation Logic; Code Coverage, Generating Coverage Reports, Using --coverage Option; Analyzing Coverage for Components and Utilities; Setting Up Coverage Thresholds in Jest Configuration.

Module 4 - Jest Intermediate Concepts : Optimizing Tests, Running Tests in Parallel, Using jest.clearAllMocks and jest.resetAllMocks to Avoid Test Contamination, Grouping Tests for Large Applications - Modular Test Suites, Dynamic Imports in Tests; Integration and E2E Testing, Writing Integration Tests for Combined Components - Testing Redux/Vuex Actions with Components, Testing HTTP Request and UI Flow; End-to-End Testing - Using Jest with Puppeteer or Playwright, Mocking User Flows and Validating Results; Error Handling in Tests - Testing Error Boundaries in React, Simulating Network Errors, Testing Fallback Components; Debugging Jest Tests, Debugging Failed Tests - Using debugger with Jest, Troubleshooting Common Issues - Timeout Errors, jsdom Environment Issues; Using Jest DevTools for Chrome/VSCode.

Module 5 – Miscellaneous : Continuous Integration and Deployment, Setting Up Jest in CI/CD Pipelines - Configuring Jest with GitHub Actions, Running Tests Automatically on Pull Requests, Automating Test Reports in CI; Performance Testing - Profiling Component Rendering Times, Testing Application Responsiveness; Testing Libraries and Tools - Using jest-extended for Custom Matchers, Integrating with React Testing Library and Vue Test Utils, Working with Storybook and Jest for Visual Testing; Best Practices for Scalable Testing - Organizing Tests for Large Projects, Folder Structure and File Naming Conventions, Writing DRY and Maintainable Tests, Balancing Unit, Integration, and E2E Tests, Avoiding Flaky Tests - Using Stable Mocks, Testing Deterministic Scenarios.

Cloud & DevOps

Module 1 - Linux Fundamentals for DevOps : What is Linux and Why Use It for DevOps?, Linux Distributions Overview (Ubuntu, CentOS, RHEL), Installing and Setting Up Linux (Locally or on Virtual Machines); Linux Basics - Basic Commands (ls, cd, cp, mv, rm, cat, touch), File and Directory Management, Viewing and Editing Files (Using nano, vim, less), User Management (adduser, usermod, passwd); Permissions and Ownership - Understanding File Permissions (chmod, chown, chgrp), File Permission Levels (Read, Write, Execute); Networking and Connectivity - Basics of SSH, Transferring Files with SCP, Checking Network Configurations (ifconfig, netstat, ping); Processes and System Monitoring - Managing Processes (ps, kill, top, htop), Scheduling Tasks with cron and at; Package Management - Installing and Updating Software (apt, yum), Using tar and zip for Compression and Archiving.

Module 2 - Cloud Computing with AWS : Introduction to AWS, Overview of Key AWS Services for DevOps, Setting Up an AWS Account; Compute Services - Working with EC2 Instances, Launching, Configuring, and Terminating EC2 Instances, Connecting to

EC2 Instances via SSH, Configuring Security Groups, Auto Scaling and Load Balancers; Storage Services - Understanding S3 (Simple Storage Service), Managing Buckets, Files, and Permissions, Using S3 for Backups; Networking in AWS - Introduction to VPC (Virtual Private Cloud), Configuring Subnets, Route Tables, and Gateways, Elastic IPs and NAT Gateways; Monitoring and Logging - Introduction to CloudWatch, Setting Up Alarms and Metrics, Using CloudTrail for Logging API Calls; IAM and Security - Understanding IAM (Identity and Access Management), Creating Users, Groups, and Roles, Implementing Policies and Permissions.

Module 3 - Continuous Integration with Jenkins : Introduction to Jenkins, Key Features and Benefits, Installing and Setting Up Jenkins (Locally and on AWS); Building Projects with Jenkins - Creating and Configuring Jenkins Jobs, Managing Pipelines, Understanding Freestyle Projects vs Pipeline Jobs; Pipeline as Code - Writing Jenkinsfiles, Declarative vs Scripted Pipelines, Using Parameters in Pipelines; Integration with Version Control - Connecting Jenkins to GitHub/GitLab, Automating Builds on Code Changes; Plugins and Extensibility - Installing and Configuring Plugins, Commonly Used Plugins for DevOps (Git, Docker, AWS); Jenkins Security - Setting Up Authentication and Authorization, Configuring Role-Based Access Control (RBAC).

Module 4 - Containerization with Docker : Introduction to Docker, Understanding Images and Containers, Installing Docker on Linux and Windows; Working with Docker Images - Pulling Images from Docker Hub, Creating Custom Docker Images, Writing and Using Dockerfiles; Managing Containers - Running and Managing Containers, Using Commands (docker run, ps, stop, rm, logs), Inspecting and Debugging Containers; Networking and Volumes - Setting Up Docker Networks, Sharing Data with Volumes and Bind Mounts, Multi-Container Networking; Docker Compose - Introduction to Docker Compose, Writing docker-compose.yml Files, Managing Multi-Container Applications; Docker Best Practices - Reducing Image Size, Securing Containers, Using Docker Bench for Security Audits.

Module 5: Orchestration with Kubernetes : Introduction to Kubernetes, Kubernetes Architecture (Nodes, Pods, Services, Controllers), Setting Up a Local Kubernetes Cluster with Minikube; Kubernetes Objects - Understanding Pods, ReplicaSets, and Deployments, Services and Networking in Kubernetes, ConfigMaps and Secrets; Managing Kubernetes Applications - Deploying Applications with kubectl, Updating Deployments and Rolling Back Changes, Scaling Applications; Advanced Kubernetes Concepts - Using Persistent Volumes and Persistent Volume Claims, Configuring Ingress Controllers for Load Balancing, Working with Namespaces for Isolation; Kubernetes with Docker - Building and Deploying Docker Images to Kubernetes, Integrating Docker Compose with Kubernetes; Monitoring and Logging - Setting Up Metrics Server, Using Prometheus and Grafana for Monitoring, Using Fluentd for Log Aggregation; Kubernetes in Production - Setting Up a Kubernetes Cluster on AWS

(EKS), Securing Kubernetes Clusters, Best Practices for Managing Large-Scale Deployments.

Material UI (Optional)

Module 1 – Basics : What is Material UI, Importance of Material Design in Modern UI Development, Setting Up Material UI, Understanding Material Design Principles, Material UI Components, Typography and Icons, Material Icons Installation and Usage, Buttons (Button Variants: contained, outlined, text, Button Groups and Icons, Customizing Buttons with Props), Layout Components, Grid System: Understanding Grid and Breakpoints, Box: Working with Box for Spacing and Layouts, Container: Managing Layout Constraints, Forms and Inputs, Text Fields: Single-Line and Multi-Line, Checkboxes, Radio Buttons, Switches, Select, Autocomplete, and Sliders.

Module 2 – Core Concepts : Input Validation with Material UI, Using FormControl and FormHelperText, Data Display Components: Cards, Lists, and Tables, Expansion Panels (Accordion), Tooltips, Chips, and Avatars, Navigation Components: App Bar, Toolbar, and Menus, Tabs and Navigation Drawers (Sidebars), Bottom Navigation, Feedback Components: Dialogs and Modals, Snackbars and Alerts, Progress Indicators (Linear and Circular), Styling in Material UI, Using the sx Prop for Inline Styling, Customizing Components with styled API.

Module 3 – Advanced Concepts : Material UI Theme Customization: Creating and Overriding Themes, Using ThemeProvider, Typography, Colors, and Spacing in Themes, Responsive Design with Material UI: Breakpoints and Responsive Layouts, Data Grid (Displaying and Managing Large Data Sets), Customizing Components (Overriding Default Styles), Using makeStyles and useStyles.

Tailwind CSS (Optional)

Module 1 – Basics : Introduction Tailwind CSS, Benefits of Using Utility-First CSS Framework, Comparison with Bootstrap and Other Frameworks, Installing Tailwind CSS: Tailwind CLI, Integrating Tailwind with React, Next.js, or Vite Projects, Tailwind CSS Basics, Understanding Utility Classes, Structure of Tailwind CSS (@tailwind base, @tailwind components, @tailwind utilities), Configuring Tailwind CSS, Tailwind Configuration File (tailwind.config.js), Adding Custom Colors, Fonts, and Spacing, Purging Unused CSS for Production Builds.

Module 2 – Core Concepts : Typography and Colors, Text Utilities: text-, font-, leading-, and tracking-, Customizing Text and Background Colors; Spacing Utilities: Margin: m-, mx-, my-, Padding: p-, px-, py-; Flexbox Utilities: flex, flex-col, justify-, items-, gap-; Grid Utilities: grid, grid-cols, gap-, col-span; Sizing and Layout: Width and Height: w-, h-, min-w-, max-h-; Box Sizing and Overflow.

Module 3 – Advanced Concepts : Responsive Design, Using Breakpoints (sm, md, lg, xl, 2xl), Mobile-First and Desktop-First Design, State Variants: Hover, Focus, Active, and Disabled States, Dark Mode with Tailwind CSS; Positioning Utilities: absolute, relative, fixed, sticky; Transform and Animation: Rotate, Scale, Skew, Transition and Duration Utilities; Customizing Tailwind: Extending Tailwind Config for Custom Utilities, Adding Plugins (e.g., Typography, Forms, Line Clamp).

Mentorship Support with Frontend Libraries & Frameworks

"Available only for those who perform well throughout the entire training program."

- **Svelte:** A modern framework that shifts much of the work to compile time, offering faster runtime and better performance for building reactive interfaces.
- **Ember.js:** A framework for ambitious web applications, offering strong conventions and powerful tools for building scalable applications.
- **Preact:** A lightweight alternative to React with similar API, ideal for high-performance, lightweight applications.
- **Backbone.js:** A library providing minimal structure to web applications with models, views, and events. Good for small projects needing basic structure.
- **Alpine.js:** A minimal JavaScript framework for handling UI interactions with declarative syntax similar to Vue.js.
- **Nuxt.js:** A framework built on Vue.js for server-side rendering, static site generation, and modern web application needs.

Useful Tools for Front-End Development

- **Visual Studio Code**: A powerful and lightweight code editor with rich extensions and integrations for JavaScript.
- **Node.js**: A runtime for executing JavaScript on the server, essential for running development tools like bundlers and package managers.
- **NPM (Node Package Manager)**: A package manager for installing and managing JavaScript libraries and dependencies.

- **Webpack**: A module bundler that optimizes JavaScript and other assets for production-ready builds.
- **Babel**: A JavaScript compiler that lets you use next-gen JavaScript features by transpiling them into browser-compatible code.
- **ESLint**: A tool for identifying and fixing issues in your JavaScript codebase by enforcing coding standards.
- **Prettier**: A code formatter that enforces consistent code style across the project.
- **Postman**: A tool for API testing and integration, often used to test RESTful services consumed by front-end applications.
- **Figma**: A collaborative design tool for creating and sharing UI/UX designs and prototypes.
- **Tailwind CSS**: A utility-first CSS framework for rapidly building custom, responsive designs directly in HTML.
- **Bootstrap**: A popular CSS framework for designing responsive and mobile-first web applications with pre-designed components.
- **Sass**: A CSS preprocessor that extends CSS with variables, nested rules, and mixins for writing maintainable styles.
- **Parcel**: A zero-config module bundler for fast and easy setup of JavaScript applications.
- **Storybook**: A tool for building UI components in isolation to document, test, and develop design systems.
- **Lighthouse**: A performance auditing tool from Google to optimize and improve the quality of web applications.
- **Axios**: A promise-based HTTP client for making API requests in JavaScript applications.
- **Chart.js**: A library for creating interactive and customizable charts and graphs in web applications.
- **Jest**: A testing framework for JavaScript applications to ensure code reliability and quality.
- **Cypress**: A testing tool for end-to-end testing of front-end applications in real browsers.
- **Vite**: A fast build tool optimized for modern JavaScript development with instant server start and hot module replacement.

How to Join the Frontend Development using JavaScript Training Program

At **Learn2Earn Labs Training Institute**, we believe in empowering aspiring frontend developers with cutting-edge skills, hands-on training, and real-world experience. To ensure the best learning outcomes, we invite applications from motivated individuals who meet the following criteria:

Eligibility Requirements

- **Educational Qualification**: A degree in a relevant domain (e.g., Computer Science, Computer Application, or any other related degree programs).
- **Passion for Learning**: Candidates must demonstrate a strong zeal to become a professional frontend developer.
- **Growth-Oriented Mindset**: We are looking for individuals who are eager to learn, adapt to new challenges, and continuously improve their skills.
- **Hard Work and Dedication**: This program is rigorous and requires commitment. Candidates must be ready to put in the effort to achieve their career goals.

How to Apply

1. Visit our website page of frontend development Training Program (two years) at **https://learntoearnlabs.com/front-end-development-course/** and read the complete details about the training program.
2. Under the **Apply Now** section, fill out the form with your details.
3. Based on your submitted details, a Learn2Earn Labs representative will contact you to further process your application or query.
4. Complete any additional steps, such as interviews, assessments, or telephonic discussions, as requested by the institute representative via email or WhatsApp.

What Happens Next?

- Once your details are reviewed, eligible candidates will be contacted for the next steps, which may include an introductory session or discussion.
- After successful enrollment, you will receive all the program details, including the schedule, enrollment ID, batch ID, terms and conditions, etc., through an enrollment confirmation letter.

Take the first step toward transforming your career! If you meet the eligibility criteria and are passionate about becoming a professional frontend developer with advanced skills, **this program is designed for you**. 😊
Join us today and pave the way to a successful career in frontend development!

LEARN 2 EARN LABS

Best Training Institute for Graduates

Live Online & Offline Classes Available

Join us for Better Career and Package Guarantee

Top Edge Training Programs

- **Full Stack Web Development**
- **Mobile App Development**
- **Front End Development**
- **Back End Development**
- **Java Full Stack**
- **Data Science & ML**
- **Digital Marketing**

Guaranteed Package (In Writing)*

- **3-5 Lakhs** (With 6 Months Training Programs)
- **5-8 Lakhs** (With 12 Months Training Programs)
- **8 Lakhs+** (With 2 Years Training Programs)

* Terms & Conditions Applied

Amenities

- Digital Notes
- Live Training Sessions
- Project Assistance
- Interview Preparation
- Working Experience
- Job Recommendations
- Professional Development
- Digital Resume & Portfolio

LEARN-2-EARN LABS TRAINING INSTITUTE

Anna Icon Complex, near Kargil Petrol Pump, Sikandra-Bodla Road, Sikandra, Agra

Website : www.LearntoearnLabs.com

Call : 91-9548868337

LEARN 2 EARN LABS

Guiding Careers with Visionary Support

Institute Director(s)

Mr. Mohit Singh
M.Tech, B.Tech (C.S.E)

Mr. Mohit Singh is a professional full-stack trainer, project consultant and startup mentor. He is holding expertise in Java, Application Design, MERN Stack, DevOps, Design Thinking and User Experience Design.
He has trained thousands of students & hundreds of employed professionals. He completed his trainings in Google, Gurugram and short-term projects in IIT Delhi, IIT BHU & IIT Jodhpur.
He is also recognized as Mentor with startup India (MAARG), Punjab Startup, startup Uttarakhand, Mumbai State Innovation Society, Atal Innovation Mission, etc. in the area of education & utility services.

https://www.linkedin.com/in/mohit9pages/

Dr. Shubhendra Gupta
Phd, B.Ed, M.Sc (Physics)

Dr. Shubhendra Gupta is an experienced digital marketer, Business Consultant and startup mentor with a demonstrated history of working in the education and services industry.
He use to train students & working professionals for getting better job opportunities and train business owners in generating profits or leads. His areas of interest are Digital Marketing, Business Development, Data Analysis, Strategic Planning, Market Research & Reality, User Testing, Website design, etc.
He is also recognized as Mentor with Startup Hubs & Innovation Labs in the area of education, brand building & business consultation.

https://www.linkedin.com/in/dmshubhendra/

Institute Vision

To be an institute that provides a transformative learning to produce highly skilled & competent professionals and to create leaders and innovators for society and industry.

LEARN-2-EARN LABS TRAINING INSTITUTE

F-4, First Floor, Anna Icon Complex,
near Kargil Petrol Pump, Sikandra-Bodla Road,
Sikandra, Agra, Uttar Pradesh, India

Website : www.LearntoearnLabs.com

Call : 91-9548868337