LEARN 2 EARN LABS

Training Syllabus

[2 Years Duration]

Full Stack Web Development

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

Full Stack Web Development Training Program (MERN Stack)

In the ever-evolving world of technology, the demand for skilled full stack developers has never been higher. Our **Full Stack Web Development Training Program** is your opportunity to master the art of building end-to-end applications and secure a future-proof career in the tech industry.

This **Two-Years Intensive Program** is meticulously designed to equip you with in-depth knowledge and hands-on experience in the most in-demand technologies. From creating visually stunning and user-friendly front-end interfaces to building powerful and scalable back-end systems, this program leaves no stone unturned in transforming you into a **job-ready full stack developer**.

About the Program

Our curriculum has been carefully crafted to ensure a balance between theoretical knowledge and real-world application. Over the course of **two years**, you will gain expertise in every aspect of full stack web development, including:

- **Front-End Development**: Start from the basics of HTML, CSS, and JavaScript, and advance to modern libraries and frameworks like ReactJS and NextJS. Learn how to create dynamic, responsive, and high-performance user interfaces.
- **Back-End Development**: Master server-side programming with Node.js, Express.js, and NestJS, and build RESTful APIs and GraphQL-powered services.
- **Database Management**: Gain proficiency in relational and non-relational databases like MongoDB, MySQL, and PostgreSQL, using tools like Mongoose and Sequelize.
- **State Management**: Learn to manage complex application states using Redux, Redux Toolkit, and RTK Query for seamless data flow across your applications.
- **DevOps Integration**: Unlock the power of modern DevOps tools like Linux, AWS, Jenkins, Docker, and Kubernetes to deploy, scale, and monitor your applications efficiently.
- **Testing and Quality Assurance**: Build quality-driven applications with Jest, React Testing Library, and end-to-end testing methodologies.

This program also incorporates **lots of project work & practical assignments**, ensuring you gain practical experience. By the end of the course, you will have built a robust portfolio that showcases your expertise and demonstrates your readiness for the professional world. **Each module is tailored to industry standards**, helping you meet the expectations of top employers worldwide.

Who is a Full Stack Developer?

A Full Stack Developer is a versatile professional who is skilled in both front-end (client-side) and back-end (server-side) development. They are capable of designing user-friendly interfaces, managing databases, handling server logic, and deploying applications to production environments. Simply put, they are the **backbone of modern web development** and one of the most sought-after roles in the tech industry.

What Makes This Program Unique?

- **Comprehensive Curriculum**: The program covers every essential technology and tool required for modern full stack development.
- **Hands-On Projects**: Work on real-world projects and assignments to apply your skills in practical scenarios.
- **Work Experience**: Based on your dedication, project submissions, and performance, you will earn **verifiable work experience**, setting you apart in the job market.
- **Expert Mentors**: Learn from industry experts who bring real-world experience and insights.
- **Career Support**: Receive guidance on building your portfolio, cracking interviews, and negotiating job offers.

Career Options After Completion

Upon completing this program, you'll be equipped to take on roles such as:

- Full Stack Developer
- Front-End Developer
- Back-End Developer
- Database Administrator
- Cloud & DevOps Engineer
- Software Engineer
- Technical Architect

With your newly acquired skills and real-world project experience, you'll stand out as a top candidate for high-paying jobs. The demand for skilled full stack developers spans across industries such as technology, e-commerce, healthcare, and finance, providing you with diverse career opportunities. You'll also have the foundation to grow into leadership roles like Team Lead or Technical Manager as you progress in your career.

Who Can Join This Program?

- **Beginners**: If you're new to programming, this program will guide you step-by-step from basics to advanced concepts.
- **Graduates and Students**: Start your career with the skills that employers are actively seeking.
- **Professionals**: Transition into the role of a full stack developer or upskill to meet industry demands.
- **Tech Enthusiasts**: Anyone passionate about creating impactful web applications can join.

Advantages of the Program

- **Future-Ready Skills**: Learn cutting-edge technologies & Concepts like React, NextJS, NodeJS, ExpressJS, NestJS, and DevOps.
- **High Earning Potential**: Full stack developers are among the highest-paid professionals in the tech industry.
- **Global Opportunities**: With these skills, you can work remotely for top companies worldwide.
- **Work Experience**: Showcase real-world experience on your resume, making you job-ready from day one.
- **Endless Career Paths**: Whether it's development, DevOps, or cloud engineering, this program opens multiple career avenues.

How Much Dedication is Required?

This is not just a course—it's a career transformation journey. Your success depends on your **dedication and commitment** to learning. By consistently participating in lectures, completing assignments, and working on real-world projects, you can:

- Build an impressive portfolio of projects.
- Earn verifiable work experience.
- Gain a competitive edge in the job market.

This journey requires focus, discipline, and a genuine passion for web development. Setting aside dedicated time for learning and staying curious will ensure you stay ahead. Every small effort you make today will contribute to your big success tomorrow. Consistency is key—small, consistent progress will always outweigh bursts of activity. Believe in your abilities and stay motivated throughout the program.

Future Scope

As the demand for web applications continues to grow, so does the need for skilled full stack developers. By completing this program, you are not only preparing for today's job market but also future-proofing your career for emerging trends in technology.

Work Experience to Boost Your Career

One of the standout features of this program is the opportunity to earn **verifiable work experience**. Your dedication to assignments, sincerity in project work, and commitment to learning will translate into real-world experience that:

- Enhances your resume.
- Increases your earning potential.
- Gives you an edge over other candidates in interviews.

This hands-on experience bridges the gap between theoretical knowledge and practical application, making you industry-ready from day one. It showcases your ability to handle real challenges and deliver impactful solutions. Employers value candidates with proven expertise, and this program ensures you have the credentials to back your skills. By the end of the program, you'll confidently step into the professional world with work experience that speaks for itself.

Transform Your Career Today

This two-year training program isn't just about learning technologies—it's about unlocking your true potential and achieving career transformation. By the end of this journey, you will:

- Be proficient in building and deploying large-scale applications.
- Have a deep understanding of modern web development practices.
- Be ready to secure high-paying roles in the tech industry.

Don't wait to build the career you've always dreamed of. Join us today and embark on a journey to become a **professional full stack developer** capable of shaping the future of technology.

Your Journey to Success Starts Here!

With our **Full Stack Web Development Training Program**, you have the tools, guidance, and opportunities to make your mark in the tech industry. The only limit is your dedication. Are you ready to transform your future? **Let's begin!** 🚀

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:
 and <hr>, HTML Comments.

Module 2 – HTML Core Concepts : HTML Text Formatting : Bold, Italic, Underline: <b>, <i>, <u>, Strong and Emphasis: , , 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.

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.

Redux, Redux Toolkit & RTK Query

Module 1 - Fundamentals of State Management : Understanding State in Applications, Types of state: Local, global, server, and UI state; Challenges in managing state in large-scale applications; Introduction to State Management Tools, Why use a state management library?, Overview of popular libraries: Redux, MobX, Context API; State Management with React's Context API, Setting up Context in a React application, Passing and consuming global state using useContext, Limitations of Context API for complex state management.

Module 2 - Core Concepts of Redux : Introduction to Redux, why is it used?, Key concepts: Store, actions, reducers, and state; Setting Up Redux - Installing Redux and setting up a basic store, Writing reducers to manage state updates, Dispatching actions to update state; Immutable State and Pure Functions, Importance of immutability in Redux, Writing pure reducers for state transformations; Debugging Redux - Installing and using Redux DevTools, Monitoring actions and state changes; Middleware in Redux, Using middleware like redux-thunk for async actions.

Module 3 - Advanced Redux and Redux Toolkit : Introduction to Redux Toolkit (RTK), Why Redux Toolkit?, Key features of RTK: Simplified setup, createSlice, and createAsyncThunk; Using createSlice for State Management - Writing slices to define

reducers and actions, Automatically generated action creators with createSlice; Async State Management with createAsyncThunk - Handling API calls using createAsyncThunk, Managing loading, success, and error states; Redux Toolkit Middleware - Using built-in middleware for logging and debugging, Adding custom middleware for side effects; Migration from Redux to Redux Toolkit - Step-by-step process to migrate an existing Redux application to RTK, Refactoring reducers and actions with slices.

Module 4 - RTK Query for Data Fetching : Introduction to RTK Query, Why use RTK Query for server-state management?, Key features: Data fetching, caching, and invalidation; Setting Up RTK Query - Adding RTK Query to an existing Redux Toolkit store, Defining API slices with createApi; Fetching and Mutating Data - Using useQuery to fetch data from APIs, Using useMutation to handle POST, PUT, DELETE requests; Automatic Caching and Invalidation - Understanding RTK Query's caching mechanism, Invalidating cache manually and automatically; Error Handling and Optimistic Updates - Handling errors in queries and mutations, Implementing optimistic updates for a better user experience.

Module 5 - State Management in Large-Scale Applications : Organizing Redux for Scalability, Structuring slices and stores for large applications., Modularizing state management logic; Best Practices for Redux and RTK - Keeping state minimal and normalized, Using selectors for efficient state access, Avoiding common anti-patterns; Integration with Full-Stack Applications - Managing front-end state with Redux while syncing with back-end APIs, Using RTK Query to handle server-side state seamlessly; Testing Redux Logic - Writing unit tests for reducers and actions, Mocking API responses for testing RTK Query logic; Performance Optimization - Avoiding unnecessary re-renders with useSelector and memorization, Using React.memo and useMemo for performance improvements.

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 –Next.js 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.

MySQL

Module 1 - MySQL Basics : Introduction to Relational Databases - What is a database, Relational vs. Non-relational Databases,Overview of MySQL, Installing MySQL, Setting up MySQL Server, Using MySQL Workbench and Command-Line Client, Database Basics - Creating and dropping databases, Data types in MySQL, Tables: Creating, altering, and dropping.

Module 2 – MySQL Core Concepts : CRUD Operations - Basic SQL Statements : INSERT, SELECT, UPDATE, DELETE, Querying Data - Filtering with WHERE, Using logical operators (AND, OR, NOT), NULL values and handling, Sorting and Limiting Results (ORDER BY, LIMIT, and OFFSET), Aggregation (COUNT, SUM, AVG, MIN, MAX), GROUP BY and HAVING clauses; Joins - INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN. Self Joins and Cross Joins; Subqueries - Inline, Correlated, and Nested Subqueries, Using subqueries in SELECT, FROM, and WHERE; Set Operations - UNION, UNION ALL, INTERSECT and EXCEPT (if supported); Window Functions - ROW_NUMBER, RANK,

DENSE_RANK, Aggregate window functions (SUM, AVG), Partitioning and ordering in window functions.

Module 3 – MySQL Advanced Concepts : Normalization - First, Second, and Third Normal Forms, Denormalization concepts; Keys and Constraints - Primary and Foreign Keys, Unique and Composite Keys, CHECK constraints; Indexing - Importance and types of indexes, Creating and managing indexes, Understanding indexing in query performance; Transactions - ACID properties, COMMIT and ROLLBACK, Savepoints; Isolation Levels - READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE,Practical examples and issues like deadlocks.

Module 4 – MySQL Intermediate Concepts : Concurrency - Locking mechanisms, Row-level and table-level locking; Writing Stored Procedures - Syntax and parameters, Using variables and flow control, Writing Functions - Scalar functions and returning values, Practical use cases; Triggers - Creating, modifying, and deleting triggers, BEFORE and AFTER triggers; User Management - Creating and managing users, Privileges and roles; Security - Securing database access, Encryption in MySQL; Performance Optimization - Query execution plans, Query optimization techniques, Monitoring and tuning MySQL performance, Backup and Restore - mysqldump and mysqlimport, Automating backups.

Module 5 - Miscellaneous : Working with Large-Scale Data, Partitioning - Horizontal and vertical partitioning, Benefits and use cases; Replication - Master-slave and master-master replication, Setting up and maintaining replication; Sharding - Understanding and implementing sharding, Challenges in sharding; Event Scheduler - Creating and managing scheduled events, Practical applications of events; Full-Text Search - Enabling and using full-text search, Fine-tuning search relevance; JSON in MySQL - Storing, querying, and manipulating JSON data, JSON functions and indexing.

MongoDB

Module 1- MongoDB Basics : Introduction to NoSQL Databases, Understanding NoSQL vs. SQL, Key features and advantages of MongoDB; Installing MongoDB - Setting up MongoDB locally, Introduction to MongoDB Atlas (Cloud Database), Using MongoDB Shell, Compass, and CLI; Understanding databases, collections, and documents; BSON structure and data types.

Module 2 – MongoDB Core Concepts : Creating and Managing Databases - Creating, renaming, and deleting databases, Viewing database statistics; CRUD Operations on

Documents - Insert operations: insertOne, insertMany, Querying documents: find, findOne, Updating documents: updateOne, updateMany, replaceOne, Deleting documents: deleteOne, deleteMany; Query Operators - Comparison operators: $eq, $ne, $gt, $lt, etc., Logical operators: $and, $or, $not, $nor, Element operators: $exists, $type, Array operators: $all, $size, $elemMatch.

Module 3 – MongoDB Advanced Concepts : Aggregation Framework - Understanding pipelines, Stages: $match, $group, $sort, $project, $limit, $skip; Aggregation expressions and operators; Handling large datasets with aggregation; Indexing - Importance of indexing, Creating and managing indexes, Compound and multikey indexes, Performance optimization with indexing; Schema Design, Schema Modeling - Embedding vs. referencing, Choosing the right schema design for performance and scalability, Working with polymorphic schemas; Data Validation - Using JSON Schema validation, Validation levels and enforcement; Relationships - One-to-one, one-to-many, many-to-many relationships, Best practices for relationship modelling.

Module 4 – MongoDB Intermediate Concepts : Understanding Transactions - Single-document atomicity, Multi-document transactions in replica sets, Practical use cases of transactions; Isolation Levels - Implementation of isolation levels in MongoDB, Handling concurrent updates and conflicts; MongoDB Administration, User and Role Management - Creating and managing users, Role-based access control (RBAC), Authentication and authorization; Backup and Restore - Using mongodump and mongorestore, Cloud backup solutions with MongoDB Atlas; Monitoring and Performance - Understanding MongoDB logs, Analyzing performance with MongoDB Compass and Atlas tools, Query performance optimization; Replica Sets - Configuring replica sets, Role of primary and secondary nodes, Failover and recovery in replica sets; Read and Write Operations, Read preferences: primary, secondary, nearest, Write concern and durability.

Module 5 - Miscellaneous : Sharding Concepts - Importance of sharding, Understanding shard keys, Configuring and managing a sharded cluster; Scaling Applications - Horizontal vs. vertical scaling, Best practices for sharding and partitioning; Full-Text Search - Creating and managing text indexes, Querying with full-text search, Handling multilingual data in text search; Working with Geospatial Data - Geospatial indexes and queries, $geoWithin and $near queries; Working with Time-Series Data - Designing schemas for time-series data, Indexing and querying time-series collections; MongoDB Atlas - Creating and managing clusters, Automated backups and scaling; Security in Atlas - Network rules and IP whitelisting, Encryption and TLS configuration; Monitoring and Alerts - Using Atlas dashboards for monitoring, Setting up alerts for performance metrics.

PostgreSQL

Module 1 - PostgreSQL Basics : Introduction to PostgreSQL, Overview of relational databases, Key features of PostgreSQL, PostgreSQL vs. other RDBMS; Installation and Setup - Installing PostgreSQL on different platforms, Introduction to pgAdmin and psql CLI, Configuring the PostgreSQL environment; Database Basics - Creating and managing databases, Understanding schemas, Exploring data types in PostgreSQL.

Module 2 – PostgreSQL Core Concepts : Data Definition Language (DDL) - Creating and modifying tables, Understanding primary keys, foreign keys, and constraints; Data Manipulation Language (DML) - INSERT, SELECT, UPDATE, DELETE operations; Querying Data - Filtering with WHERE, Using logical operators (AND, OR, NOT), Sorting and limiting results (ORDER BY, LIMIT, OFFSET); Aggregation and Grouping - COUNT, SUM, AVG, MIN, MAX, GROUP BY and HAVING clauses; Joins - INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, Self joins and cross joins; Subqueries - Inline, correlated, and nested subqueries, Subqueries in SELECT, FROM, and WHERE; Common Table Expressions (CTEs) - Writing and using CTEs, Recursive CTEs; Window Functions - ROW_NUMBER, RANK, DENSE_RANK, Aggregate window functions (SUM, AVG), Partitioning and ordering in window functions.

Module 3 - PostgreSQL Advanced Concepts : Normalization and Denormalization - Understanding 1NF, 2NF, 3NF, and BCNF, When to denormalize for performance; Constraints - PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY constraints and cascading actions; Indexing - Creating and managing indexes, B-Tree, GIN, GiST, and BRIN indexes, Performance optimization with indexing; Transactions - ACID properties, COMMIT and ROLLBACK, Savepoints; Isolation Levels - READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE, Handling deadlocks and conflicts; Concurrency Control - Locking mechanisms, Table-level and row-level locks; Writing Stored Procedures - Syntax and parameters, Using procedural languages (PL/pgSQL); Writing Functions - Scalar and table-returning functions, IMMUTABLE, STABLE, and VOLATILE functions; Triggers - BEFORE and AFTER triggers, Practical use cases of triggers.

Module 4 - PostgreSQL Intermediate Concepts : Full-Text Search - Configuring and querying full-text search, Using tsvector and tsquery, Indexing for full-text search; JSON and JSONB - Storing and querying JSON data, JSON operators and functions, Indexing JSON fields; Geospatial Data - Using PostGIS for geospatial queries, Geometric and geographic data types; User and Role Management - Creating and managing roles, Granting and revoking privileges; Security - Authentication methods, Configuring SSL for secure connections; Backup and Restore - Using pg_dump and pg_restore, Automating backups; Monitoring and Performance Tuning -

Understanding query plans (EXPLAIN and EXPLAIN ANALYZE), Using pg_stat views for performance analysis, Optimizing queries and database performance.

Module 5 - Miscellaneous : High Availability and Scalability, Replication - Setting up streaming replication, Logical replication and publication/subscription; Partitioning - Table partitioning and use cases, Performance benefits of partitioning; Clustering and Load Balancing - Understanding clustering, Setting up load balancers for PostgreSQL; Understanding Extensions - Installing and managing extensions, Popular PostgreSQL extensions (PostGIS, hstore, etc.); Writing Custom Extensions - Basics of extension development.

Node.js

Module 1 – Node.js Basics : Introduction to Node.js, Key Features of Node.js: Asynchronous, Event-Driven Architecture, Advantages of Node.js for Backend Development, Understanding the Node.js Runtime Environment; Node.js Architecture - Single-Threaded Event Loop, Non-Blocking I/O Model, Comparing Node.js with Traditional Backend Technologies; Setting Up Node.js - Installing Node.js and npm, Using nvm for Managing Node.js Versions, Writing and Running Your First Node.js Script.

Module 2 – Node.js Core Concepts : Global Objects: global, __dirname, __filename, Using the process Object: Accessing Environment Variables and Command-Line Arguments, Understanding Node.js Timers: setTimeout, setInterval, setImmediate; Basic Debugging in Node.js - Using console for Logging, Debugging with Node.js Debugger; Node.js Modules, Built-in Modules - Overview of Core Modules (fs, path, os, http, crypto, etc.), Understanding the Role of CommonJS in Node.js; Creating and Managing Custom Modules - Writing and Exporting Custom Functions and Classes, Importing and Using Custom Modules; Working with npm - Installing and Managing Dependencies, Creating and Publishing npm Packages; Event-Driven Programming, Understanding Events in Node.js - Event-Driven Architecture Overview, Using the events Module; Custom Event Emitters - Creating and Listening to Events, Chaining Events and Handling Errors; Practical Use Cases of Event Emitters - Real-Time Notifications, Modular Event-Driven Design.

Module 3 – Node.js Advanced Concepts : File System Operations, File System Module - Reading and Writing Files (fs.readFile, fs.writeFile), Appending, Renaming, and Deleting Files, Creating and Managing Directories; Asynchronous and

Synchronous File Operations - Differences and Use Cases for Both Approaches; Practical Applications - Building a Basic File Uploader, Watching File Changes with fs.watch; Introduction to Streams - Types of Streams: Readable, Writable, Duplex, and Transform, Understanding the Stream Lifecycle; Working with Buffers - Creating and Manipulating Buffers, Encoding and Decoding Binary Data; Practical Use Cases - Streaming Large Files, Implementing File Uploads and Downloads; Creating HTTP Servers, Understanding the http Module - Setting Up Basic HTTP Servers, Handling HTTP Methods (GET, POST, PUT, DELETE); Working with Request and Response Objects - Parsing Query Strings and Request Bodies, Setting Response Headers and Status Codes; Serving Static Content - Handling Static Files Without Frameworks.

Module 4 – Node.js Intermediate Concepts : Routing Basics - Implementing Basic Routing Logic, Handling Dynamic Routes and URL Parameters; Creating Modular Routing Systems - Organizing Routes in Separate Files and Modules, Error Handling for Undefined Routes; HTTPS and Secure Connections, Setting Up HTTPS Servers - Configuring SSL/TLS Certificates, Handling Secure HTTP Connections (https Module), Best Practices for Secure Communication - Enforcing HTTPS in Applications, Preventing Common Vulnerabilities (e.g., Man-in-the-Middle Attacks); Practical Use Cases - Building a Secure API Gateway; Database Integration, Relational Database Integration - Setting Up MySQL or PostgreSQL Databases, Connecting Using mysql2 or pg Modules, Writing and Executing SQL Queries; NoSQL Database Integration - Connecting to MongoDB Using the mongodb Driver, Designing and Querying Collections; Practical Database Applications - Building User Management Systems, Creating CRUD APIs for Data Management.

Module 5 - Miscellaneous : Authentication and Security, User Authentication - Storing and Validating User Credentials, Implementing Login and Registration Systems; Secure Password Storage - Hashing Passwords with bcrypt, Implementing Password Reset Functionality; Data Security Best Practices - Protecting Against Injection Attacks, Using Environment Variables for Configuration; Performance Optimization - Understanding Event Loop Delays, Using Worker Threads for CPU-Intensive Tasks; Caching - Implementing Basic In-Memory Caching, Optimizing Repeated Database Queries; Monitoring and Debugging - Using Tools Like pm2 for Process Management, Debugging Node.js Applications with VS Code.

Express.js

Module 1 - Express.js Basics : Introduction to Node.js Environment for Express.js, Basics of the Node.js runtime and its importance for Express.js, Introduction to Express.js, The Role of Express.js in Backend Development, Differences Between Express.js and Other Backend Frameworks, Installing Node.js and npm, Setting Up a Basic Express.js Project, Using express Module and Running the First App, Understanding Express.js Project Structure - Organizing Files and Folders, Setting Up Configuration Files and Environment Variables; Basic Routing - Defining Routes for HTTP Methods (GET, POST, PUT, DELETE), Understanding req and res Objects, Handling Query Parameters and Route Parameters; Handling Asynchronous Code with Promises and Async/Await in Express.js; Basic Request Validation for Routes.

Module 2 – Express.js Core Concepts : Static File Serving - Using express.static() to Serve Static Assets, Managing Static File Caching and Compression; Express.js Debugging - Setting Up Debug Logs with the debug Module, Monitoring Incoming Requests and Outgoing Responses; Advanced Routing - Nested and Conditional Routes, Organizing Routes in Modular Files Using express.Router(), Route Groups and Middleware Prioritization; Middleware in Express.js - Types of Middleware: Built-in, Third-Party, and Custom Middleware, Writing Advanced Middleware: Logging, Request Transformation, and Error Handling, Middleware Chaining and Execution Order; Practical Middleware Applications - Authentication Middleware for JWT and Sessions, Input Validation Middleware Using express-validator; File Upload Handling - Using multer for Single/Multiple File Uploads, Validating File Types and Sizes, Storing Files Locally or in Cloud Services (e.g., AWS S3).

Module 3 – Express.js Advanced Concepts : Introduction to Template Engines, Benefits of Using Template Engines in Backend Applications, Comparison of Popular Template Engines (Pug, EJS, Handlebars), Setting Up Template Engines in Express.js, Rendering HTML Templates with Dynamic Data, Using Layouts, Partials, and Helpers; RESTful API Design, Principles of REST API Design and Best Practices, Structuring Endpoints and Versioning APIs, Building a Complete CRUD API, Implementing CRUD Operations for Resources, Handling Pagination, Sorting, and Filtering, Integrating Database Connectivity for CRUD with - MySQL: Using mysql2 module for connections, MongoDB: Using mongoose for database operations, PostgreSQL: Using pg module for queries; Error Handling in APIs - Centralized Error-Handling Middleware for APIs, Returning Proper Status Codes and Messages, Custom Error Classes and Standard Error Response Formats; Manual API Testing - Using Tools Like Postman and Insomnia, Monitoring API Logs for Performance Bottlenecks, Using Custom Error Logging Middleware; Rate Limiting and Abuse Prevention - Implementing Request Limits with Libraries like express-rate-limit.

Module 4 – Express.js Intermediate Concepts : Authentication Mechanisms - Session-Based Authentication, Token-Based Authentication with JWT; JWT Basics, Understanding JWT Structure: Header, Payload, and Signature, Benefits of Stateless Authentication, Practical JWT Use Cases - Generating JWTs for Authenticated Users, Verifying Tokens and Protecting Endpoints; Advanced JWT Features - Implementing Refresh Tokens, Setting Token Expiry and Revocation; Authorization Mechanisms - Implementing Role-Based Access Control (RBAC), Permission-Based Access Control (PBAC), Integrating Middleware for Role Validation; Best Practices - Securing Login and Registration Routes, Protecting Sensitive Endpoints; Advanced Database Queries for Authentication - Storing and Retrieving Encrypted Credentials in MySQL, MongoDB, and PostgreSQL, Implementing Forgot Password and Password Reset Features.

Module 5 – Miscellaneous : Session Management - Using express-session for Session Handling, Storing Session Data Securely; Cookies in Express.js - Setting, Reading, and Managing Cookies with req.cookies and res.cookie(), Secure Cookies with Flags (httpOnly, secure, sameSite); Persistent Login Systems - Combining Cookies and Sessions for Authentication; Local Storage and Security - Using Local Storage for Temporary Data Management, Combining Local Storage with Backend Logic; Security Best Practices - Securing APIs Against XSS, CSRF, and SQL Injection, Using Helmet.js and Other Middleware for Secure Headers, Validating Inputs with express-validator, Implementing CORS Policies with cors Middleware; Performance Optimization - Reducing Middleware Overhead, Optimizing Static File Delivery with Compression, Implementing Lazy Loading for Middleware; Monitoring and Profiling - Tools for Monitoring Application Performance, Debugging and Optimizing Slow Endpoints, Using APM Tools (Application Performance Monitoring) Like New Relic; Caching in Express.js - Implementing Basic In-Memory Caching, Using Cache-Control Headers for Static Content, Redis-Based Caching for Dynamic Content, Deployment Strategies, Deploying Express.js Applications on Cloud Providers (AWS), Using Docker for Containerized Deployments, Implementing CI/CD Pipelines for Automated Deployment.

Sequelize

Module 1 - Sequelize Basics : Introduction to Sequelize, Benefits of using Sequelize with MySQL, Comparing Sequelize with other ORMs; Setting Up Sequelize - Installing Sequelize and Sequelize CLI, Installing MySQL Drivers, Setting up a MySQL Database, Initializing Sequelize in an Express.js Project; Understanding Sequelize Workflow - Models, Migrations, and Seeds, How Sequelize Interacts with MySQL, Connection Configuration (sequelize.config.js).

Module 2 - Sequelize Core Concepts : Defining Models - Creating Models using Sequelize CLI, Understanding Fields, Data Types, and Options, Defining Primary Keys, Auto Increment, and Default Values, Using allowNull and Validations; CRUD Operations - Creating Records (create), Reading Records (findAll, findOne, findByPk), Updating Records (update), Deleting Records (destroy); Querying Data - Using where Clause for Filtering, Operators (Op.eq, Op.gt, Op.like, etc.), Ordering and Pagination (order, limit, offset).

Module 3 - Sequelize Advanced Concepts : Associations in Sequelize; Types of Relationships: One-to-One, One-to-Many, Many-to-Many; Defining Associations in Models (belongsTo, hasOne, hasMany, belongsToMany) - Fetching Related Data with include, Lazy and Eager Loading; Migrations and Seeders - Understanding Migrations in Sequelize, Creating and Running Migrations, Adding/Removing Columns, Using Seeders to Populate Initial Data; Model Scopes - Defining Scopes for Query Reusability, Default Scopes vs Custom Scopes.

Module 4 - Sequelize Advanced Concepts : Transactions - Understanding Transactions in Sequelize, Implementing Managed and Unmanaged Transactions, Error Handling with Transactions; Hooks (Lifecycle Events) - Using Hooks for Pre- and Post-Operations, Hooks: beforeCreate, afterUpdate, beforeDestroy, etc., Practical Use Cases for Hooks; Complex Queries - Using Raw Queries with Sequelize, Joins and Aggregations, Subqueries in Sequelize; Performance Optimization - Query Optimization Techniques, Using Indexes for Faster Queries, Logging and Debugging Queries.

Module 5 – Miscellaneous : Integrating Sequelize with Express.js - Setting up Sequelize in an Express.js Project, Structuring Models, Controllers, and Routes, Using Repositories or Services to Handle Sequelize Operations; Building RESTful APIs - Creating Endpoints for CRUD Operations, Handling Associations in APIs (Nested Resources), Error Handling and Validation in Sequelize; Authentication and Authorization - User Authentication with Sequelize and JWT, Role-Based Access Control (RBAC); Testing and Deployment - Writing Unit Tests for Sequelize Models, Testing Endpoints with Mock Databases.

Mongoose

Module 1 – Mongoose Basics : Introduction to Mongoose, Why Use It?, Advantages of Mongoose Over Native MongoDB Driver, Understanding MongoDB and NoSQL Basics; Setting Up the Development Environment - Installing MongoDB Locally or Using MongoDB Atlas, Installing Mongoose in a Node.js Project, Connecting Mongoose to MongoDB, Understanding Mongoose Connection Options.

Module 2 – Mongoose Core Concepts : Mongoose Schema vs. Model, Defining Your First Schema, Creating and Using a Model, Writing Basic CRUD Operations; Database Design with Mongoose - Structuring Collections and Documents, Mapping JavaScript Objects to MongoDB Documents,Best Practices for Designing MongoDB Collections; Debugging and Monitoring - Handling Connection Errors, Using Mongoose Debug Mode for Query Logging, Monitoring MongoDB Performance Using Compass; Schema Basics - Defining Fields in a Schema, Understanding Schema Types (String, Number, Date, Array, etc.), Required, Default, and Validation Properties; Custom Validations - Creating Schema-Level Custom Validators, Validating Data Formats (e.g., Email, Phone Numbers), Using Built-In Validators and Custom Error Messages; Indexes - Adding Indexes to Schema Fields, Compound Indexes for Multiple Fields, Using Sparse and Unique Indexes; Model Methods - Writing Instance Methods for Documents, Creating Static Methods for Models; Using Middleware (pre, post) for Lifecycle Events; Populating Data - Referencing Other Collections, Using populate to Fetch Related Data, Populating Nested Documents.

Module 3 - Mongoose Advanced Concepts : Relationships - Modeling One-to-One, One-to-Many, and Many-to-Many Relationships, Embedding vs. Referencing Data, Managing Data Integrity Between Related Collections; Aggregations - Introduction to Aggregation Pipelines, Common Stages ($match, $group, $sort, $project), Using Aggregations for Complex Data Analysis; Virtual Fields - Adding Virtual Properties to Documents, Using Virtuals for Computed Properties, Populating Virtuals; Middleware and Hooks - Understanding Pre and Post Middleware, Using Middleware for Logging, Validation, and Transformations, Handling save, update, and remove Middleware; Error Handling - Managing Validation Errors, Handling Unique Constraint Violations, Using Mongoose Error Messages in Express.js Responses.

Module 4 - Mongoose Intermediate Concepts: Query Optimization - Efficient Querying with select and lean, Using Indexes to Optimize Performance, Query Caching with Redis and Mongoose; Pagination and Sorting - Implementing Pagination with skip and limit, Adding Sorting Options in Queries, Building Advanced Query Filters; Transactions - Introduction to MongoDB Transactions, Using Sessions with Mongoose, Managing Multi-Document Transactions; Data Encryption and Security - Storing Sensitive Data Securely, Hashing Passwords with Mongoose and bcrypt, Implementing

Field-Level Encryption; Advanced Use Cases - Using Mongoose for File Storage Metadata, Implementing Real-Time Features with Change Streams, Managing Large Datasets with Sharding.

Module 5 – Miscellaneous : Testing with Mongoose - Writing Unit Tests for Models and Schema Validation, Using Mock Databases (mongodb-memory-server) for Testing, Integration Testing with Mongoose Queries; Deploying Mongoose Applications - Connecting to MongoDB Atlas for Production, Managing Environment-Specific Configurations, Handling Backups and Data Recovery; Error Reporting and Debugging - Logging Queries for Debugging, Using Mongoose Debug Mode in Production; Best Practices for Mongoose - Structuring Models in Large Applications, Optimizing Schema Design for Scalability, Using Plugins for Common Functionalities; Integrating Mongoose with Express.js - Using Mongoose Models in Express.js Controllers, Error Handling Middleware for Mongoose Validation Errors, Building Reusable Services for Database Operations.

GraphQL

Module 1 - GraphQL Basics : Introduction to GraphQL, How is it different from REST APIs?, Core concepts: Schema, Query, Mutation, Subscription, Advantages of GraphQL over REST, Setting up the environment for GraphQL development; Setting Up a Basic GraphQL Server - Installing dependencies (express, graphql, express-graphql, etc.), Creating a simple GraphQL server, Defining the first schema and resolver, Testing with GraphQL Playground or Apollo Sandbox.

Module 2 - GraphQL Core Concepts : GraphQL Schema Design - Types, Fields, and Resolvers, Query and Mutation basics, Input types for mutations, Enum and custom scalar types, Aliases, Fragments, and Variables in queries; Advanced Schema Design - Nested queries and mutations, Handling relationships between types, Interfaces and Unions, Directives in GraphQL.

Module 3 - GraphQL Advanced Concepts : Working with MySQL - Setting up a MySQL , database and connecting with Node.js, Writing resolvers to fetch data from MySQL, Using query builders like Knex.js or ORM like Sequelize, Managing relationships (One-to-Many, Many-to-Many) in GraphQL, Pagination and filtering with GraphQL and MySQL; Working with MongoDB - Setting up MongoDB and connecting with Node.js, Writing resolvers to fetch data from MongoDB, Querying nested documents with GraphQL, Pagination and filtering with MongoDB, Differences in resolver handling between MySQL and MongoDB; Apollo Server Integration -

Introduction to Apollo Server, Setting up Apollo Server with Express, Migrating from express-graphql to Apollo Server, Using Apollo Client to test queries and mutations.

Module 4 – GraphQL Intermediate Concepts : Error Handling in GraphQL- Understanding and managing errors in resolvers, Custom error messages and error codes, Handling validation errors, Logging errors with tools like Winston or Morgan; Authentication and Authorization - Authenticating users with JWT, Protecting queries and mutations with middleware, Role-based access control in GraphQL, Field-level authorization; Subscriptions in GraphQL - Introduction to real-time features in GraphQL, Setting up WebSocket for subscriptions, Using GraphQL Subscriptions to handle real-time updates, Example use case: Chat application.

Module 5 – Miscellaneous : Optimizing GraphQL Performance - Batching and caching with DataLoader, Preventing over-fetching and under-fetching, Limiting query depth and complexity, Using persisted queries; Advanced Features and Best Practices - Schema stitching and federation for microservices, Modularizing schema and resolvers, Testing GraphQL APIs with Jest or Mocha, Securing GraphQL endpoints against common vulnerabilities, Documentation with tools like GraphQL Voyager or GraphQL Docs.

Redis

Module 1 - Redis Basics : Introduction to Redis, Overview of Redis as an In-Memory Data Store, Key Features and Use Cases, Redis vs. Traditional Databases; Installing and Setting Up Redis - Installing Redis on Local Machines (Linux, macOS, Windows), Setting Up Redis on Cloud Platforms (AWS Elasticache, Azure, GCP), Configuring Redis for Local and Remote Access; Basic Redis Commands - Working with Keys (SET, GET, DEL, EXISTS, TYPE), Expiring Keys with EXPIRE and TTL, Working with Strings (SET, GET, APPEND, INCR, DECR), Using Redis CLI for Command Execution; Connecting Redis with Express.js - Installing Redis Client Libraries (e.g., ioredis, redis), Establishing Connection to Redis, Basic Operations: Reading and Writing Data from Redis; Monitoring and Debugging - Using Redis Logs and Monitoring Tools, Managing Connections and Checking Redis Status with INFO.

Module 2 – Redis Core Concepts : Working with Redis Data Structures, Strings - Storing and Manipulating String Values, Use Cases: Caching, Token Storage; Lists - Adding and Removing Items (LPUSH, RPUSH, LPOP, RPOP), Retrieving Data with LRANGE, Use Cases: Queues and Logs; Hashes - Creating and Accessing Fields (HSET, HGET, HGETALL), Updating and Deleting Fields, Use Cases: Storing User Profiles; Sets -

Adding and Removing Members (SADD, SREM, SMEMBERS), Checking Membership with SISMEMBER, Use Cases: Tags, Unique Elements; Sorted Sets - Adding Members with Scores (ZADD, ZRANGE, ZREM), Use Cases: Leaderboards and Rankings; Advanced Commands for Data Structures - Iterating with SCAN, HSCAN, SSCAN, ZSCAN, Using MULTI for Atomic Transactions with Data Structures.

Module 3 - Redis Advanced Concepts : Introduction to Caching - Why Use Redis for Caching, Benefits of In-Memory Caching for Express.js Applications; Implementing Caching in Express.js - Setting Up Middleware for Caching, Storing API Responses in Redis, Using TTL for Expiring Cache Entries; Cache Strategies - Cache Aside, Write-Through, Read-Through; Advanced Caching - Implementing Hierarchical Caching, Managing Cache Invalidation, Avoiding Cache Stampede with Locking Mechanisms; Debugging Cache Issues - Identifying Stale or Expired Data, Using Redis Commands (KEYS, TTL, MONITOR) for Troubleshooting.

Module 4 - Redis Advanced Features : Pub/Sub for Real-Time Messaging - Understanding the Publish/Subscribe Pattern, Implementing Pub/Sub with Redis in Express.js, Use Cases: Notifications, Chat Applications; Redis Streams - Introduction to Redis Streams, Creating and Consuming Streams (XADD, XREAD, XDEL), Managing Stream Groups for Scalability; Redis Transactions - Using MULTI and EXEC for Atomic Transactions, Handling Errors and Rollbacks; Redis Persistence - Configuring Snapshotting (RDB), Using Append-Only File (AOF) for Data Durability, Best Practices for Balancing Performance and Persistence; Rate Limiting and Throttling - Implementing Rate Limiting for APIs, Using Redis to Throttle User Requests, Preventing Abuse with Key Expiry and Incremental Counters; Redis Security - Securing Redis Instances with Passwords, Using SSL for Secure Connections, Setting Up Access Control Lists (ACLs).

Module 5 – Miscellaneous : Scaling Redis - Understanding Redis Clustering, Setting Up a Redis Cluster for High Availability, Configuring Redis Sentinel for Failover; Redis Performance Optimization - Using PIPELINE for Batch Processing, Optimizing Queries with Indexing Patterns, Monitoring Performance with Redis Profiler Tools; Using Redis in Microservices - Managing Shared States Across Microservices with Redis, Implementing Distributed Locks with SETNX; Redis in Production - Deploying Redis with Docker, Hosting Redis on Cloud Platforms (AWS Elasticache), Managing Backups and Recovery; Combining Redis with Other Databases (MongoDB, PostgreSQL) for Hybrid Architectures.

NestJS

Module 1 – NestJS Basics : Overview of the NestJS Framework, Benefits of Using NestJS in Backend Development, Comparison with Other Backend Frameworks (e.g., Express.js, Spring Boot); Setting Up the Development Environment - Installing Node.js and npm, Installing the NestJS CLI (Command-Line Interface), Generating the First NestJS Application,Understanding the Default Project Structure, Running a NestJS Application in Development Mode; Core Concepts and Components : Controllers - Creating Basic Controllers, Handling Route Parameters, Defining Routes for HTTP Methods (GET, POST, PUT, DELETE), Using Query Parameters and Request Body; Services - Creating Basic Services, Injecting Services into Controllers, Encompassing Business-Domain Logic in Services; Modules - Creating Modules, Managing Application Structure with Modules, Understanding Module Encapsulation.

Module 2 – NestJS Core Concepts : REST API Development Basics - Creating a REST API Application, Handling Update and Delete Requests, Implementing Pagination with Query Parameters, Handling Malicious Request Data, Sending User-Friendly Error Messages, Setting Response Status Codes; Tools and Utilities - Installing and Using Insomnia or Postman for API Testing, Debugging Applications with NestJS Logger, Using Hot Module Replacement (HMR) for Development Efficiency; Advanced Application Structure and Dependency Management; Data Transfer Objects (DTOs) - Introduction to DTOs, Validating Input Data with class-validator, Auto-transform Payloads to DTO Instances with class-transformer; Dependency Injection (DI) - Core Concepts of Dependency Injection, Exploring NestJS Providers - Value-Based Providers, Non-Class-Based Provider Tokens, Class Providers, Factory Providers, Async Providers; Creating Custom Providers; Scope Management in Providers - Controlling Provider Scope (Singleton vs Request-Scoped), Diving Deeper into Request-Scoped Providers; Application Configuration - Introducing the @nestjs/config Module, Customizing Environment File Paths, Schema Validation for Configuration, Using the Config Service for Dynamic Configuration, Custom Configuration Files, Configuration Namespaces and Partial Registration, Asynchronously Configuring Modules; Dynamic Modules - Understanding Dynamic Modules in NestJS, Creating and Using Dynamic Modules, Leverage Dynamic Modules for Scalable Applications.

Module 3 – NestJS Advanced Concepts : Filters, Guards, and Interceptors, Exception Filters - Catching Exceptions with Built-in and Custom Filters, Returning User-Friendly Error Responses; Guards - Protecting Routes with Guards, Using Metadata to Build Generic Guards, Implementing Role-Based and Permission-Based Guards; Interceptors - Adding Pointcuts with Interceptors, Handling Timeouts with Interceptors, Transforming Responses with Interceptors; Middleware and Decorators - Using Built-in Middleware, Creating Custom Middleware, Adding Request Logging with

Middleware, Managing Cross-Origin Resource Sharing (CORS); Custom Param Decorators - Creating and Using Custom Decorators; OpenAPI and Swagger - Introducing the Swagger Module, Generating OpenAPI Specifications, Using CLI Plugins for Swagger, Decorating Model Properties, Adding Example Responses, Using Tags to Group Resources; Request Lifecycle and Logging - Understanding the Request Lifecycle in NestJS, Advanced Logging Techniques with Middleware, Integrating Third-Party Logging Libraries (e.g., Winston, Pino).

Module 4 – NestJS Intermediate Concepts : Database Integration, Setting Up Database Connections, MySQL with TypeORM - Connecting NestJS to MySQL, Defining Entities and Repositories, Performing CRUD Operations; MongoDB with Mongoose - Connecting NestJS to MongoDB, Defining Schemas and Models, Aggregations and Advanced Queries; PostgreSQL with Prisma - Setting Up Prisma with NestJS, Defining Models and Relationships, Writing Queries and Handling Transactions; Authentication and Authorization, Implementing JWT Authentication - Understanding JWT Structure, Generating and Validating Tokens, Protecting Routes with Auth Guards; Role-Based Access Control (RBAC) - Managing Roles and Permissions, Implementing Role-Based Guards, Secure Session Management, Using express-session for Persistent Logins; Security Best Practices - Securing Applications Against XSS, CSRF, and SQL Injection, Using Helmet.js for Secure HTTP Headers, Configuring CORS for Secure Cross-Origin Requests.

Module 5 - Miscellaneous : GraphQL Integration - Setting Up GraphQL with @nestjs/graphql, Defining Schemas and Resolvers, Using Apollo Server with NestJS, Advanced GraphQL Features - Queries, Mutations, and Subscriptions, Managing Complex Relationships; Microservices with NestJS - Introduction to Microservices Architecture, Setting Up a Microservices Application, Using Message Brokers (e.g., RabbitMQ, Kafka), Communication Patterns: Request/Response and Event-Based, Handling Failures and Retries in Microservices; Testing - Unit Testing, Writing Tests for Controllers, Services, and Modules, Using Mocks and Spies; Integration Testing - Testing Database Interactions, Simulating HTTP Requests with Supertest; End-to-End Testing with Jest; Deployment and Scaling - Building Production-Ready Applications, Using Docker to Containerize NestJS Applications, Deploying on Cloud Providers (AWS), Configuring Load Balancers and Horizontal Scaling; Additional Tools - Event Emitters in NestJS, Cache Management with @nestjs/cache, Task Scheduling with @nestjs/schedule.

Jest

Module 1 - Introduction to Jest : Introduction to Jest, Key features of Jest: Snapshot testing, built-in mocking, asynchronous testing; Jest's role in the JavaScript testing ecosystem, Use cases for Jest in full-stack development; Installing Jest globally and locally, Setting up Jest in a React project, Setting up Jest for a Node.js application, Configuring jest.config.js for custom setups: Test environments (jsdom, node), Transformations for TypeScript and Babel; Basic Concepts - Writing your first test case: Syntax and conventions, The anatomy of a Jest test: describe, test, and it, Writing and running test suites, Organizing tests in the project structure: Co-located vs. dedicated test folders; Matchers for Assertions - Basic matchers: toBe, toEqual, toMatch, toBeNull, toBeDefined, Numeric matchers: toBeGreaterThan, toBeLessThan, String and array matchers: toContain, toMatchObject.

Module 2 - Testing Front-End Applications with Jest : React Component Testing, Unit testing for functional components: Testing props, state, and default values, Testing class components: Verifying lifecycle methods with mocks; Snapshot Testing - Generating snapshots for React components, Updating and maintaining snapshots, Ignoring dynamic data (e.g., timestamps) in snapshots; Mocking in Jest - Mocking functions and methods using jest.fn(), Mocking React child components and third-party libraries, Mocking network requests using jest.mock and axios-mock-adapter; Testing Asynchronous React Code - Testing component updates triggered by promises and async/await, Using Jest's fake timers (jest.useFakeTimers) to test timeouts and intervals; Integration with React Testing Library - Rendering components for tests with render(), Testing user interactions: Simulating clicks, inputs, and form submissions, Verifying DOM changes triggered by state updates.

Module 3 - Testing Back-End Applications with Jest : Setting Up Jest for Back-End Applications - Installing Jest in a Node.js/Express project, Writing test cases for plain JavaScript functions and modules; Testing APIs with Supertest - Setting up Supertest for HTTP request testing, Writing tests for RESTful APIs: Validating request payloads and responses, Checking status codes, headers, and data; Mocking Database Operations - Mocking Mongoose models: Simulating database queries like find, save, delete; Using in-memory MongoDB for realistic tests: Setting up mongodb-memory-server, Writing integration tests with in-memory databases; Testing Middleware Logic - Writing unit tests for custom middleware: Authentication middleware, Validation middleware; Handling Authentication in Tests - Testing JWT-based authentication: Verifying token validation and protected routes, Mocking authenticated requests with custom headers.

Module 4 - Advanced Jest Features : Custom Matchers, Writing custom matchers for reusable assertions, Extending Jest's matchers with custom utilities; Test Coverage Reporting - Enabling test coverage in jest.config.js, Interpreting coverage metrics: Functions, lines, statements, branches; Improving coverage for critical application areas; Mocking Timers and Dates - Using Jest's fake timers for time-dependent logic,

Mocking date functions like Date.now() with jest.spyOn; Parameterized Testing - Writing tests for multiple input sets using test.each, Creating data-driven test cases for edge scenarios; Debugging Jest Tests - Debugging failed tests with console.log and breakpoints, Using --watch mode to rerun specific tests.

Module 5 - Testing Full-Stack Applications : End-to-End Testing (E2E), Setting up Jest for full-stack testing, Writing tests for complete workflows: User signup and login, Creating, retrieving, and deleting resources; Mock Services for Integration Tests - Using mock servers for testing API interactions, Testing third-party APIs using mocks and spies; Testing Data Validation - Writing tests for schema validation in back-end services, Ensuring proper error messages and HTTP status codes for invalid payloads; CI/CD Integration - Running Jest tests in CI/CD pipelines: Using GitHub Actions to automate test execution, Generating and storing coverage reports; Best Practices for Full-Stack Testing - Writing tests for maximum maintainability: Organizing test data and mocks, Balancing unit, integration, and E2E tests, Writing descriptive test names and assertions for readability.

DevOps for Full Stack Web Development

Module 1 - Linux for DevOps : Introduction to Linux and its role in DevOps, Linux file system structure and navigation, Basic commands: File operations (ls, cd, cp, mv, rm), permissions (chmod, chown); User and Group Management - Managing users (adduser, deluser) and groups (groupadd, groupdel), File and directory permissions; Process and Service Management - Managing processes: ps, top, kill, Starting, stopping, and managing services (systemctl, service); Shell Scripting - Basics of shell scripting (bash, sh), Writing and executing scripts, Automating repetitive tasks using shell scripts; Networking and Security - Basic networking commands: ifconfig, ping, netstat, curl; SSH: Secure file transfer and remote server management, Setting up a basic firewall with iptables.

Module 2 - AWS for DevOps : Overview of cloud computing and AWS services, Setting up an AWS account and understanding the AWS Management Console; Compute Services - Launching and managing EC2 instances, Configuring security groups and key pairs, Elastic Load Balancer (ELB) and Auto Scaling basics; Storage and Database Services - S3: Creating buckets, managing objects, and permissions; RDS: Setting up and connecting to managed databases, Basics of DynamoDB for NoSQL databases; Infrastructure as Code (IaC) - Introduction to AWS CloudFormation, Writing basic CloudFormation templates; Monitoring and Logging - Setting up CloudWatch for metrics and alerts, Viewing and analyzing logs using AWS CloudTrail.

Module 3 - Jenkins for CI/CD : What is Jenkins and its role in CI/CD?, Setting up Jenkins on a server, Jenkins pipeline architecture; Creating Jenkins Pipelines - Writing and managing Declarative Pipelines, Configuring Jenkins jobs to pull code from GitHub, Automating builds, tests, and deployments; Integration with Build Tools - Using Jenkins with npm and yarn for front-end builds, Automating back-end builds and tests; Integrating Jenkins with AWS - Deploying applications to AWS from Jenkins pipelines, Automating EC2 instance creation and S3 uploads; Jenkins Plugins - Must-have plugins for DevOps pipelines: GitHub Integration, Docker Build and Publish, Kubernetes Continuous Deploy.

Module 4 - Docker for Containerization : Introduction to Docker, Understanding containers and Docker's role in DevOps, Installing Docker and understanding Docker components (docker, docker-compose); Working with Docker Images and Containers - Pulling and running containers from Docker Hub, Building custom Docker images using Dockerfiles, Managing containers: Starting, stopping, and removing; Docker Compose - Writing docker-compose.yml files for multi-container applications, Running React/Next.js, Node.js, and database containers together; Networking and Volumes - Creating and managing Docker networks, Using Docker volumes for persistent storage; Docker Registry - Pushing images to Docker Hub, Setting up a private Docker registry.

Module 5 - Kubernetes for Orchestration : Introduction to Kubernetes, Understanding container orchestration and Kubernetes architecture, Setting up a local Kubernetes cluster using Minikube, Understanding pods, nodes, and clusters; Kubernetes Resources - Deployments: Writing deployment.yaml files, Services: Exposing applications with ClusterIP, NodePort, and LoadBalancer, ConfigMaps and Secrets: Managing configuration and sensitive data; Scaling and Rolling Updates - Horizontal Pod Autoscaler (HPA), Rolling updates and rollback strategies for deployments; Helm for Kubernetes - Introduction to Helm and its role in managing Kubernetes applications, Installing and managing Helm charts; Monitoring and Logging in Kubernetes - Setting up monitoring with Prometheus and Grafana, Viewing logs with kubectl logs and Fluentd.

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.

- **Fastify:** A fast and low-overhead framework designed for high-performance applications.
- **Koa.js:** Created by the same team behind Express.js, Koa provides a more modern and modular approach to web application development.
- **Meteor.js:** A full-stack framework for building web and mobile applications, featuring real-time updates and a seamless integration with MongoDB.

Popular Tools for Front-End and Back-End Development Using JavaScript

Front-End Development Tools

- **Webpack:** A module bundler for optimizing and managing front-end assets and dependencies.
- **Vite:** A fast build tool and development server optimized for modern web projects.
- **Tailwind CSS:** A utility-first CSS framework for creating customizable and responsive designs.
- **Bootstrap:** A popular CSS framework for quickly building responsive, mobile-first websites.
- **Parcel:** A zero-configuration build tool for front-end applications.
- **Storybook:** A UI development tool for creating, testing, and documenting components in isolation.

Back-End Development Tools

- **GraphQL:** A query language and runtime for building APIs with flexible data-fetching capabilities.
- **Apollo Server:** A GraphQL server for building scalable APIs with tools for managing schema and queries.
- **Mongoose:** An ODM (Object Data Modeling) library for MongoDB and Node.js.
- **Sequelize:** A promise-based ORM (Object-Relational Mapper) for Node.js and SQL databases.
- **PM2:** A process manager for running and managing Node.js applications in production.
- **Prisma:** A modern database toolkit for working with SQL and NoSQL databases in JavaScript.
- **JSON Web Token (JWT):** A standard for securely transmitting information as a JSON object for authentication.
- **Redis:** An in-memory data store used for caching and session management in Node.js applications.

How to Join the Full Stack Web Development Training Program

At **Learn2Earn Labs Training Institute**, we believe in nurturing talent and shaping the future of aspiring full stack web developer using MERN. To ensure the best outcomes for our students, we welcome applications from 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 backend developer.
- **Growth-Oriented Mindset:** We are looking for individuals who are eager to learn, embrace challenges, and continuously improve their skills.
- **Hard Work & Dedication:** This program is intensive and requires commitment. Candidates must be ready to put in the effort to master backend development.

How to Apply

1. Visit our website page of backend development Training Program (two years) at **https://learntoearnlabs.com/full-stack-web-development-course-mern-24-months/** 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 full stack web developer with advanced skills, **this program is designed for you**. 😊
Join us today and pave the way to a successful career in full stack web 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