Training Syllabus

[2 years Duration]

Backend Development with JavaScript

Join us for Better Career and Job Guarantee

[Live Online & Offline Classes Available]

LEARN-2-EARN LABS TRAINING INSTITUTE, AGRA

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

Backend Development Training Program with JavaScript

In the fast-paced world of technology, backend development forms the backbone of every successful application. As industries evolve, the demand for proficient backend developers who can meet high standards and deliver efficient, scalable solutions is growing exponentially. At **Learn2Earn Labs Training Institute**, we are committed to bridging the gap between aspiring developers and the industry by offering a **Comprehensive Backend Development Training Program with JavaScript** designed to empower you with cutting-edge knowledge and hands-on experience.

This training program is not just a course; it's your gateway to a thriving career in backend development. From mastering foundational concepts to gaining expertise in the latest tools and technologies, our program equips you with the skills to stand out in the highly competitive tech world. Whether you're aiming to secure a high-paying job, build impactful projects, or simply grow as a developer, this program is your definitive solution.

A Curriculum Built for Excellence

Our syllabus has been meticulously crafted after thorough research into the current and future demands of the tech industry. It dives deep into the latest trends, technologies, and practices, ensuring that you are always a step ahead. The program covers a wide spectrum of tools and technologies, including:

- **Core Technologies**: JavaScript, TypeScript, JSON, Asynchronous JavaScript, Git & GitHub.
- **Database Mastery**: MySQL, MongoDB, PostgreSQL.
- **Backend Frameworks and Libraries**: Node.js, Socket.IO, Express.js, GraphQL, WebSockets, NestJS.
- **Advanced Tools**: Redis, TypeORM, Mongoose, Prisma.
- **Testing Frameworks**: Jest, Mocha.
- **DevOps Skills**: Jenkins, Linux, AWS, Docker, Kubernetes.
- **Optional Topics**: Kafka, RabbitMQ, ZooKeeper (for those eager to explore more).

This robust curriculum ensures you are industry-ready and well-prepared to tackle real-world challenges with confidence.

Hands-On Learning and Project Experience

Learning by doing is the cornerstone of our approach. Throughout the program, you will work on real-world projects and assignments that reflect actual industry scenarios. These projects will not only solidify your knowledge but also provide you with a portfolio to showcase during interviews.

From building scalable APIs to implementing advanced authentication systems, from integrating databases to optimizing application performance, you will gain hands-on experience in every crucial aspect of backend development.

Additionally, based on your performance, contributions, and project submissions, you may qualify for **work experience opportunities** (subject to terms and conditions). This unique advantage allows you to demonstrate practical skills and experience to potential employers, giving you an edge in the job market.

Why Choose Backend Development Training Program?

1. **Industry-Relevant Skills**: The program focuses on the latest trends and tools, ensuring you are job-ready from day one.
2. **Comprehensive Learning**: With in-depth coverage of every critical backend development concept, you'll gain a solid foundation to build a successful career.
3. **Expert Trainers**: Learn from seasoned professionals with years of industry experience who are committed to your success.
4. **Career-Focused Approach**: The program is designed to help you secure top-notch placements by meeting the rigorous demands of today's tech industry.
5. **Flexibility and Customization**: Optional topics like Kafka, RabbitMQ, and ZooKeeper allow you to explore additional areas of interest and deepen your expertise.

Transform Your Career

In today's competitive environment, just being good isn't enough—you need to be exceptional. This program not only teaches you the technical skills required to excel but also instills the confidence and mindset to solve complex problems, innovate, and grow.

By the end of this training, you won't just be a backend developer; you'll be a **professional backend developer** capable of creating impactful solutions and thriving in any competitive environment.

Unlock Your Potential

This training program is more than a learning experience; it’s an investment in your future. It’s an opportunity to acquire skills, gain confidence, and build a career you’re proud of. The knowledge and projects you gain here will not only prepare you for job interviews but also help you succeed in real-world work environments.

Don’t wait for opportunities to come your way—create them! Join our training program and embark on a journey that will transform your career and your life.

For more information, visit our website **www.LearntoEarnLabs.com**, or contact us at **+91-9548868337**. Let us help you unlock your true potential and achieve your career dreams!

Your journey to becoming a professional backend developer starts here.

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.

Data Structure using JavaScript

Module 1 - Fundamentals of Data Structures : Introduction to Data Structures, Why are they needed in JavaScript applications?, Overview of Data Structures: Linear, Non-Linear, Dynamic, and Static, Applications of Data Structures in real-world software development; Algorithmic Foundations : Time Complexity: Constant, Linear, Quadratic, Logarithmic; Space Complexity, Big-O Notation: Best-case, worst-case, average-case, Understanding Recursion: Basics, Stack usage, Recursive vs Iterative solutions; Mathematical Foundations : Modular arithmetic and its applications in hashing, Number Theory (LCM, GCD, prime numbers, etc.) for coding problems.

Module 2 - Linear Data Structures : Arrays - Operations: Insert, Delete, Update, Advanced Array Techniques: Prefix Sum, Sliding Window, Two-pointer technique; Rotations and Merging: Efficient implementations; Sorting in Arrays: Bubble, Selection, Insertion, Merge, Quick, and Heap sort; Searching in Arrays: Linear and Binary search; Strings as Data Structures;Character Arrays and Mutable Strings; Pattern Matching Algorithms: Naive String Matching, KMP Algorithm (Knuth-Morris-Pratt), Boyer-Moore and Rabin-Karp algorithms; Linked Lists - Singly Linked List: Node creation, Insertion, Deletion, Reversal, Doubly Linked List: Bidirectional Traversal, Memory Optimization, Circular Linked List: Applications in gaming and simulations; Stacks - LIFO Principle: Push, Pop, Peek operations, Stack implementation using Arrays and Linked Lists, Applications: Balancing Parentheses, Postfix/Prefix Evaluations; Queues - FIFO Principle, Implementing Simple, Circular, and Double-Ended Queues, Priority Queues and Real-world Applications; Deque - Double-ended Queues and their use cases, Sliding Window Maximum Problem.

Module 3 - Non-Linear Data Structures: Trees - Tree Terminology: Root, Leaf, Depth, Height, etc., Binary Tree: Insertion, Deletion, Traversals (DFS - Pre, In, Post; BFS - Level Order), Binary Search Tree: Properties, Operations, Balancing, AVL Trees: Rotations and Rebalancing; Heaps: Min-Heap and Max-Heap, Applications in Priority Queues and Heap Sort; Tries (Prefix Trees) - Structure and Representation, Operations: Insert, Search, Delete, Applications: Autocomplete, Dictionary implementation; Graphs - Representation: Adjacency Matrix, Adjacency List, Types: Directed, Undirected, Weighted, Traversal Techniques: BFS, DFS, Shortest Path Algorithms: Dijkstra's Algorithm, Bellman-Ford Algorithm, Floyd-Warshall Algorithm, Applications: Social Networks, Web Crawling, Network Routing; Disjoint Sets (Union-Find) - Concepts of Union and Find, Path Compression and Union by Rank, Applications: Kruskal's MST Algorithm.

Module 4 - Advanced Data Structures : Hashing - Hash Tables: Implementation and Collision Handling (Chaining, Open Addressing), Applications: Caching, Anagrams Check, Frequency Counters; Segment Trees - Tree Representation for Range Queries, Lazy Propagation for Range Updates; Fenwick Trees (Binary Indexed Trees) - Efficient Range Queries, Applications in Competitive Programming; Graphs: Advanced Algorithms - Minimum Spanning Tree: Prim's and Kruskal's Algorithms, Strongly Connected Components: Tarjan’s Algorithm, Topological Sorting; Ternary Search Trees - Efficient Dictionary Lookups, Auto-suggestions and Spell Correction.

Module 5 - Algorithmic Paradigms : Divide and Conquer - Principles and Examples: Merge Sort, Quick Sort, Applications in Binary Search; Dynamic Programming - Memoization vs Tabulation, Key Problems: Fibonacci Sequence, Longest Common Subsequence,Matrix Chain Multiplication, Knapsack Problem; Greedy Algorithms - Principles and Applications: Activity Selection, Huffman Coding; Backtracking - Problem-solving approach, Key Problems: N-Queens, Sudoku Solver; Bit Manipulation - Binary Representations, Efficient Techniques for XOR, AND, OR, Applications: Subset Generation.

TypeScript

Module 1 - TypeScript Basics : Introduction to TypeScript, Key features and advantages over JavaScript, Installing and setting up TypeScript (Node.js, npm, and ts-node), Compiling TypeScript into JavaScript; TypeScript Fundamentals : Type annotations: string, number, boolean, any, unknown, Type inference and Type assertions (as keyword), Union and Intersection types, Arrays and Tuples, Enums.

Module 2 - TypeScript Core Concepts : Functions in TypeScript - Function types and return types, Optional and default parameters, Rest parameters, Arrow functions; Object-Oriented Programming with TypeScript - Classes and interfaces, Access modifiers (public, private, protected), Static properties and methods, Abstract classes and methods, Inheritance and method overriding; Modules and Namespaces - Export and Import keywords, Using ES modules in TypeScript, Namespaces for logical grouping.

Module 3 - TypeScript Advanced Concepts : Advanced Type Features - Generics: Functions, classes, and interfaces, Utility types: Partial, Readonly, Pick, Omit, Record, Mapped types and conditional types, Template Literal Types; Error Handling in TypeScript - Handling exceptions with try, catch, finally, Custom error classes, Type-safe error handling patterns; Decorators in TypeScript - Introduction to decorators, Class, method, and property decorators, Practical use cases: Logging, validation, and dependency injection; TypeScript Configuration - Understanding tsconfig.json, Compiler options (strict, noImplicitAny, module, target, etc.), Working with source maps for debugging.

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.

Asynchronous JavaScript

Module 1 - Foundations of Asynchronous JavaScript : Introduction to Asynchronous JavaScript, Synchronous vs. Asynchronous Execution, Event Loop, Call Stack, and Task Queue, Blocking vs. Non-Blocking Code, Use Cases of Asynchronous Programming in Backend Development; Understanding Callbacks - Definition and Use Cases of Callbacks, Implementing Callback Functions, Error-First Callbacks, Callback Hell: Issues and Real-World Examples.

Module 2 - Promises : Introduction to Promises, The Promise Object and its States (Pending, Resolved, Rejected), Creating Promises: new Promise(), Resolving and Rejecting Promises, Promise Methods - then(), catch(), and finally(), Chaining Promises, Handling Errors in Promises; Advanced Concepts - Using Promise.all, Promise.race, Promise.allSettled, and Promise.any, Common Patterns in Promises for Backend Development.

Module 3 - Async-Await : Introduction to Async-Await, Syntax and Usage in Backend Applications, Converting Promises into Async-Await; Error Handling with Async-Await - Using try-catch for Error Management, Combining Async-Await with finally(); Advanced Async-Await Patterns - Sequential vs. Parallel Execution of Async Functions, Avoiding Deadlocks and Infinite Waits; Practical Use Cases - Database Queries, File Handling.

Module 4 - Fetch API - Introduction to Fetch API, Fetching Resources from Servers, Fetch API vs. XMLHttpRequest; Using Fetch API - fetch(), Handling Responses with Response.json() and Response.text(), Error Handling with Fetch; Advanced Usage - Setting Custom Headers, Making POST, PUT, DELETE Requests, Uploading and Downloading Files; Practical Use Cases - Fetching Data from REST APIs, Consuming APIs in Microservices.

Module 5 - Axios - Introduction to Axios, Why Use Axios? Comparison with Fetch API, Setting Up Axios in Backend Projects; Using Axios for API Requests - Syntax and Basic GET/POST Requests, Sending Data with POST, PUT, DELETE Methods; Axios Configuration - Setting Global Defaults, Creating Axios Instances for Modular API Calls; Error Handling in Axios - Using try-catch with Axios, Interceptors for Request and Response Error Handling; Advanced Axios Features - Canceling Requests with Axios, Concurrent Requests with axios.all, Handling Timeouts and Retry Logic.

Module 6 - Miscellaneous : Concurrency Management - Parallel vs. Sequential Execution, Using Promise.all and Async Iterators; Streaming Data - Consuming Server-Sent Events (SSE), Handling Streams in Fetch API; Integrating Asynchronous Functions in Backend - Querying Databases Asynchronously, Consuming External APIs in Microservices Architecture; Debugging Asynchronous Code - Common Pitfalls and How to Avoid Them, Using Developer Tools for Debugging Asynchronous Code.

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.

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.

Socket.io

Module 1: Socket.io Basics – Introduction to Socket.IO, Overview and Key Features, Socket.IO vs. WebSockets, Use Cases in Backend Development, Installing Socket.IO with npm, Understanding Socket.IO Architecture (Client and Server); Creating a Simple Node.js Server, Integrating Socket.IO into the Server.

Module 2 – Socket.io Core Concepts : Serving Static HTML Files, Setting Up a Basic HTML Client, Linking the Socket.IO Client Library; Building a Web Interface - Adding Basic Interactivity with HTML, CSS, and JavaScript, Establishing Real-Time Communication Between Client and Server; Integrating Socket.IO with Backend Logic - Connecting the Client to the Server (io.connect), Handling Client-Server Communication Events; Bidirectional Communication - Sending Data from Client to Server (socket.emit), Receiving Data from Server to Client (socket.on); Real-Time Application Scenarios - Chat Application Integration, Live Notifications.

Module 3 – Socket.io Advanced Concepts : Emitting Events, Custom Events - Creating and Emitting Custom Events, Handling Events on Client and Server; Advanced Event Handling - Sending Complex Data Structures (JSON, Arrays),Event Propagation Across Multiple Clients; Acknowledgments - Implementing Event Acknowledgments for Reliable Communication; Introduction to Broadcasting, Difference Between Emitting and Broadcasting Events; Server-Side Broadcasting - Broadcasting Events to All Connected Clients, Using the socket.broadcast Method for Specific Clients; Use Cases of Broadcasting - Announcing New Connections, Real-Time Updates for All Users.

Module 4 – Socket.io Intermediate Concepts : Core API Methods - Server Methods: io.emit, io.on, io.to, Client Methods: socket.emit, socket.on; Advanced API Usage - Working with Rooms and Namespaces, Managing Event Lifecycles; Handling Disconnections, Understanding Disconnections - Reasons for Client Disconnections, Detecting Disconnection Events (disconnect); Graceful Disconnection Handling - Cleaning Up Resources, Informing Other Clients About Disconnections; Use Cases - Removing Users from Active Lists, Reassigning Roles or Resources; Connection State Recovery, Handling Reconnections - Understanding the connect_error and reconnect

Events, Implementing Automatic Reconnection Logic; Restoring Connection State - Preserving User Data Across Reconnections, Strategies for Resuming Sessions; Practical Scenarios - Rejoining Rooms After Reconnection, Synchronizing Lost Messages.

Module 5 - Miscellaneous : Server Delivery, Ensuring Reliable Event Delivery - Understanding Event Delivery Mechanisms in Socket.IO, Using Event Acknowledgments for Delivery Assurance; Handling Server Failures - Implementing Fallback Mechanisms, Ensuring Continuity in Real-Time Communication; Client Delivery, Client-Side Event Management - Tracking Event Responses, Resending Failed Events Automatically; Optimizing Delivery for Slow Connections - Using Compression for Data Efficiency, Minimizing Client-Side Latency, Scaling Horizontally, Introduction to Scaling Socket.IO Applications - Challenges of Scaling Real-Time Systems, Benefits of Horizontal Scaling; Redis Adapter for Scaling - Setting Up Redis Adapter for Multi-Server Communication, Broadcasting Events Across Servers; Load Balancing for Socket.IO - Configuring Load Balancers (NGINX, HAProxy), Session Affinity and Sticky Sessions; Monitoring and Debugging Scaled Applications - Tools for Monitoring Socket.IO Performance, Analyzing Logs and Resolving Scalability Issues.

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.

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.

WebSockets

Module 1 - WebSocket Basics : Introduction to WebSockets, Overview of WebSocket Protocol, Comparison with HTTP and REST APIs, Full-Duplex Communication Explained; Advantages of WebSockets in Backend Development - Low Latency Communication, Real-Time Data Streaming, Key Use Cases in Backend Applications; Setting Up WebSockets - Installing and Configuring WebSocket Libraries (e.g., ws), Understanding WebSocket Handshakes.

Module 2 – WebSocket Core Concepts : Establishing Connections - Creating a Basic WebSocket Server, Initiating Client Connections to the Server; Communication in WebSockets - Sending and Receiving Messages Between Client and Server, Handling Text and Binary Data; Event Handling - Key WebSocket Events: open, message, error, close, Writing Handlers for Each Event.

Module 3 - WebSocket Advanced Concepts : Message Formats and Protocols - Using JSON for Structured Messaging, Designing Custom Messaging Protocols; Broadcasting and Multicasting - Sending Messages to Multiple Clients, Efficient Broadcasting Techniques; Binary Data Handling - Streaming Media and Large Files, Working with Buffers in WebSocket Communication; Handling Disconnections and Reconnections, Detecting Disconnections - Understanding WebSocket Close Events (close Codes), Handling Unexpected Client or Server Disconnections; Implementing Reconnection Logic - Auto-Reconnection Strategies for Clients, Preserving State Across Reconnections; Use Cases - Managing Active User Sessions, Resuming Real-Time Feeds After Network Interruptions.

Module 4 - WebSocket Intermediate Concepts : WebSocket Security, Authentication with WebSockets - Securing Initial Handshakes with JWT or API Keys, Authenticating Messages Post-Handshake; Data Encryption - Using Secure WebSockets (wss://), Ensuring End-to-End Encryption for Messages; Preventing Common WebSocket Vulnerabilities - Protecting Against Cross-Site WebSocket Hijacking, Implementing Rate Limiting and IP Blocking ; Scaling WebSocket Applications, Horizontal Scaling with WebSockets - Challenges of Scaling WebSocket Servers, Using Load Balancers with Sticky Sessions; Pub/Sub for WebSocket Scalability - Integrating Pub/Sub Mechanisms (e.g., Kafka, Redis Pub/Sub), Distributing Messages Across Multiple Servers; Monitoring and Debugging Scaled WebSocket Systems - Analyzing Traffic and Latency Metrics, Debugging Real-Time Communication Issues.

Module 5 - Miscellaneous : Integration with Backend Systems, Database Integration - Streaming Real-Time Updates from Databases, Using WebSockets to Push Notifications for Data Changes; File Uploads and Media Streaming - Implementing Chunked Uploads Over WebSockets, Live Media Streaming in Backend Applications;

Microservices Communication - Connecting WebSocket Services with REST and RPC APIs, Performance Optimization, Reducing Latency - Optimizing Message Payloads, Using Compression for WebSocket Data; Efficient Connection Management - Handling High Client Loads with Connection Pools, Configuring Server Timeouts and Keep-Alive Mechanisms.

TypeORM

Module 1 - TypeORM Basics : Overview of Object-Relational Mapping (ORM), Features and Benefits of TypeORM, Comparison with Other ORMs (e.g., Sequelize, Mongoose), Prerequisites for Using TypeORM with Node.js; Setting Up the Development Environment - Installing Node.js and npm, Installing TypeORM and Required Dependencies (typeorm, reflect-metadata, and database drivers), Creating a New TypeORM Project, Configuring the ormconfig.json/ormconfig.ts File, Setting Up a Basic Database (MySQL/PostgreSQL/SQLite).

Module 2 - TypeORM Core Concepts : Understanding Entities, Introduction to Repositories, What are Migrations, and Why Are They Needed?, Connection Management in TypeORM; Building the First Application - Defining a Simple Entity, Performing Basic CRUD Operations Using Repositories, Running the Application with TypeORM CLI; Entities and Relationships - Creating Entities, Using Decorators for Entity Definitions (@Entity, @Column, @PrimaryGeneratedColumn), Column Types and Options (e.g., unique, nullable, default), Establishing Relationships, One-to-One, One-to-Many, and Many-to-Many Relationships, Using @JoinColumn and @JoinTable, Eager and Lazy Loading; Query Building - Using TypeORM Repositories for CRUD, Writing Custom Queries Using QueryBuilder, SELECT, INSERT, UPDATE, and DELETE Queries, Adding Where Clauses and Joins, Using QueryBuilder for Complex Queries; Indexes and Constraints - Defining Indexes in Entities, Using Unique Constraints, Composite Primary Keys and Foreign Keys; Advanced Entity Features - Soft Deletes and Restoration, Version Control and Optimistic Locking, Working with Embedded Entities; Error Handling - Catching and Handling TypeORM-Specific Errors, Database Connection and Query Error Scenarios.

Module 3 - TypeORM Advanced Concepts : Advanced Repository and Manager Usage - Custom Repository vs. Default Repository, Using Entity Manager for Advanced Queries, Using Transactional Entity Manager; Migrations - Creating Migrations Using TypeORM CLI, Applying Migrations to Update Schema, Reverting Migrations, Writing Custom SQL Queries in Migrations; Database Initialization - Auto-Synchronization vs. Migrations, Handling Schema Changes in Production; Event Listeners and Subscribers - Using @BeforeInsert, @AfterInsert, @BeforeUpdate, @AfterUpdate,

@BeforeRemove, and @AfterRemove Decorators, Creating and Using Entity Subscribers; Custom Repository Patterns - Implementing Custom Methods in Repositories, Extending the Default Repository, Reusing Query Logic Across Repositories.

Module 4 - TypeORM Intermediate Concepts : Performance Optimization and Advanced Patterns, Caching in TypeORM - Introduction to Query Caching, Configuring and Using Cache Options in TypeORM, Cache Expiry and Invalidations; Pagination and Filtering - Implementing Pagination with skip and take, Adding Sorting and Filtering in Queries, Handling Large Datasets Efficiently; Database Transactions - Transaction Management in TypeORM, Implementing Nested Transactions, Using Isolation Levels (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE), Handling Deadlocks and Rollbacks; Design Patterns with TypeORM - Data Mapper vs. Active Record Patterns, Implementing the Repository Pattern, Applying the Unit of Work Pattern; Working with Large Datasets - Using Streaming for Large Queries, Batch Inserts and Updates, Optimizing Bulk Operations.

Module 5 - Miscellaneous : Testing with TypeORM - Unit Testing with In-Memory Databases (e.g., SQLite), Integration Testing with Mock Databases, Writing Tests for Repositories and Custom Methods, Using Fixtures for Test Data; Deployment Best Practices - Preparing TypeORM Applications for Production, Handling Database Connections in Production, Managing Environment-Specific Configurations; Database Integrations - Connecting TypeORM with Popular Databases, MySQL: Configuration, Optimization, and Best Practices, PostgreSQL: Using JSONB Columns, Window Functions, and Extensions, SQLite: Lightweight Database for Development and Testing, MongoDB (Using TypeORM MongoDB Integration), Using Multiple Database Connections; Security Best Practices - Preventing SQL Injection in Queries, Securing Database Credentials with Environment Variables, Using Role-Based Access Control (RBAC) in Queries; Real-World Integrations - Combining TypeORM with GraphQL Resolvers, Using TypeORM in a Microservices Architecture, Integration with Redis for Caching.

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.

Prisma

Modules 1 – Prisma Basics : Overview of Prisma as an ORM, Benefits of Using Prisma for Database Management, Prisma vs. Other ORMs (TypeORM, Sequelize, Mongoose); Setting Up the Development Environment - Installing Node.js and npm, Installing Prisma CLI and Required Dependencies, Setting Up a New Prisma Project, Initializing Prisma with prisma init, Setting Up the .env File for Database Configuration, Understanding the Prisma Schema, Datasource, Generator, and Model Blocks; Database Setup - Connecting to a Relational Database (PostgreSQL, MySQL, SQLite), Running Initial Database Migrations, Managing the Prisma Client.

Module 2 – Prisma Core Concepts : Defining Models in Prisma Schema, Understanding Prisma Types and Data Mappings, Performing Basic CRUD Operations Using Prisma Client; Tools and Utilities - Installing and Using Prisma Studio, Managing Data with Prisma Studio, Debugging Prisma Applications with Logging; Schema Modeling and Query Operations, Modeling Relationships - One-to-One Relationships, One-to-Many Relationships, Many-to-Many Relationships, Managing Referential Integrity with Cascades; Advanced Model Features - Using @id, @default, and @unique Attributes, Auto-Incremented Fields, Defining Composite Keys and Indexes, Modeling Enums and Embedded Documents; CRUD Operations, Using Prisma Client for - create, findUnique, findMany, update, delete, Advanced Query Features (Filters, Pagination, Sorting); Batch Operations - Bulk Inserts and Updates; Aggregations with Prisma Client - count, avg, sum, min, max; Query Optimization - Selecting Specific Fields (select and include), Using Nested Queries, Lazy vs. Eager Loading in Relationships; Data Validation and Constraints - Enforcing Constraints in Schema, Validating User Input at the Database Level.

Module 3 – Prisma Advanced Concepts : Prisma Migrations - Understanding the Migration Workflow, Creating and Applying Migrations with prisma migrate, Rolling Back Migrations, Managing Schema Changes in Production; Advanced Query Techniques - Handling Complex Joins with Prisma Client, Using Raw SQL Queries with Prisma, Combining Raw SQL with Prisma Queries; Database Seeding - Writing Seed

Scripts for Initial Data, Automating Seeding in Development; Event Hooks - Introduction to Prisma Middleware, Creating and Using Lifecycle Hooks (before, after), Practical Use Cases for Middleware (Logging, Validations); Multi-Datasource Support - Connecting Prisma to Multiple Databases, Managing Multiple Datasources in a Single Project.

Module 4 – Prisma Intermediate Concepts : GraphQL Integration - Setting Up Prisma with Apollo Server, Generating Resolvers with Prisma Client, Writing Queries, Mutations, and Subscriptions, Managing Relationships in GraphQL with Prisma; REST API Integration - Building a REST API with Prisma and Express.js, Handling CRUD Operations with Prisma Client, Implementing Pagination, Sorting, and Filtering in APIs; Authentication and Authorization - Implementing Role-Based Access Control (RBAC), Managing Users, Roles, and Permissions with Prisma, Using Prisma Middleware for Access Control; Working with Cloud Databases - Connecting Prisma to Cloud Databases (AWS RDS), Using Prisma with Supabase and PlanetScale; Performance Optimization - Optimizing Queries with Indexing, Managing Large Datasets with Pagination, Query Optimization Best Practices.

Module 5 - Miscellaneous : Testing with Prisma - Writing Unit Tests for Prisma Queries, Using Mock Databases for Testing, Integration Testing with Prisma Client; Deployment Best Practices - Preparing Prisma Applications for Production, Managing Environment-Specific Configurations, Optimizing Connection Pools with Prisma, Using Docker to Deploy Prisma-Based Applications; Advanced Features - Working with JSON Fields in Prisma, Using Prisma with MongoDB (Beta Features), Implementing Soft Deletes and Versioning, Event-Driven Architecture with Prisma; Monitoring and Debugging - Logging Queries with Prisma Client, Integrating Prisma with APM Tools; Task Scheduling - Implementing Scheduled Jobs with Prisma and Node.js.

Jest

Module 1 - Getting Started with Jest : Introduction to Jest, Why Use It?, Advantages of Jest for Node.js and Express.js Applications, Comparing Jest with Other Testing Frameworks (Mocha, Jasmine); Setting Up the Development Environment - Installing Jest in a Node.js Project, Configuring Jest in package.json or with a jest.config.js File, Understanding Test File Naming Conventions, Running Tests Using the Jest CLI.

Module 2 – Jest Core Concepts : Core Concepts - Test Suites and Test Cases, Understanding describe and test Blocks, Writing Your First Jest Test, Basic Matchers (toBe, toEqual, toContain, etc.); Using Watch Mode - Running Tests with --watch for Real-Time Feedback, Filtering Tests by Name or Filename; Debugging Jest Tests - Using console.log and Debugging Tools, Interpreting Jest Error Messages; Testing Functions and Modules - Writing Unit Tests for Pure Functions, Testing Asynchronous Functions with async/await and .resolves/.rejects, Mocking Timers and Delays with jest.useFakeTimers; Mocking and Stubbing - Using Jest Mocks for Dependency Isolation, Mocking External Modules and APIs, Creating Manual Mocks, Spies and Tracking Function Calls (jest.fn(), jest.spyOn); Testing Express.js Controllers - Writing Tests for Route Handlers, Mocking req, res, and next Objects, Testing HTTP Status Codes and JSON Responses; Testing Middleware - Creating Test Cases for Custom Middleware, Ensuring Proper Execution Flow in Middleware; Error Handling - Testing Error Scenarios, Validating Responses for Different Error Codes.

Module 3 - Jest Advanced Concepts : Snapshot Testing - Creating Snapshots for Objects and UI Components, Updating and Verifying Snapshots, Use Cases for Snapshot Testing in APIs; Integration Testing - Writing Integration Tests for Express.js Routes, Testing End-to-End Request-Response Cycles, Mocking Database Connections (e.g., MongoDB, PostgreSQL); Testing Asynchronous Code - Using Mock Promises for Simulating Async Operations, Testing API Calls with Jest, Handling Race Conditions in Tests; Database Mocking - Using Libraries Like mockingoose for MongoDB, Mocking Sequelize/TypeORM Queries, Testing CRUD Operations Without Affecting the Actual Database; Advanced Matchers - Deep Equality Matching, Testing with Regular Expressions and Arrays, Validating Complex Objects and Nested Properties.

Module 4 - Jest Advanced Concepts : Advanced Mocking Techniques - Mocking Classes and Instances, Mocking Files and Modules Dynamically, Resetting and Restoring Mocks Between Tests; Testing Performance - Measuring Execution Time of Functions, Ensuring Code Meets Performance Benchmarks; Code Coverage - Enabling Code Coverage in Jest, Analyzing Coverage Reports, Improving Test Coverage for Node.js and Express.js Applications; Testing APIs - Mocking HTTP Calls with Libraries Like nock, Writing Tests for RESTful APIs, Testing Pagination, Sorting, and Filtering Endpoints; Advanced Error Handling - Simulating Failures in External Services, Validating Fallback Mechanisms in Tests.

Module 5 - Miscellaneous : Integration with CI/CD - Setting Up Jest in Continuous Integration Pipelines (GitHub Actions, Jenkins, etc.), Running Tests Automatically on Pull Requests, Using Jest with Dockerized Applications; Testing for Security - Validating Authentication and Authorization Mechanisms, Testing JWT Token Validation and Middleware, Ensuring Role-Based Access Control (RBAC) Works as Expected; Testing Express.js with Databases - Setting Up Test Databases for Integration Testing; Error Reporting - Using Custom Test Reporters with Jest, Exporting Test Results in JSON/HTML Format; Best Practices - Writing Maintainable and Readable Tests, Avoiding Flaky Tests and Ensuring Test Consistency, Organizing Tests in Large Applications, Leveraging Jest Extensions and Plugins.

Mocha

Module 1 - Mocha Basics : Overview of Mocha as a Testing Framework, Key Features and Benefits, Comparison with Other Testing Frameworks (Jest), Mocha's Role in Testing Node.js and Express.js Applications; Setting Up the Development Environment - Installing Mocha Globally and Locally, Setting Up Mocha in a Node.js Project, Writing and Running Your First Test Case, Using the Mocha CLI to Run Tests.

Module 2 - Mocha Core Concepts : Understanding Test Suites and Test Cases, Writing Tests with describe and it Blocks, Using Hooks (before, after, beforeEach, afterEach) for Setup and Teardown, Basic Assertions - Introduction to Assertion Libraries (Chai, Node.js assert), Using Common Assertions (equal, deepEqual, ok, etc.), Handling Test Failures and Debugging; Mocha Configuration - Setting Up Custom Test Directories and File Naming, Configuring Timeout for Tests; Testing Functions and Modules - Writing Unit Tests for Pure Functions, Handling Synchronous and Asynchronous Code, Testing with Callbacks, Promises, and async/await; Testing Express.js Controllers - Writing Tests for Route Handlers, Mocking req, res, and next Objects, Validating HTTP Status Codes and Response Payloads; Testing Middleware - Testing Custom Middleware, Mocking and Validating Middleware Execution Flow; Mocking and Stubbing - Introduction to Sinon.js for Mocking and Stubbing, Creating Spies and Tracking Function Calls, Mocking External Services and APIs; Error Handling - Testing Error Responses and Status Codes, Simulating Failures in Express.js Routes.

Module 3 – Mocha Advanced Concepts : Integration Testing - Testing Express.js API Endpoints, Simulating HTTP Requests with Supertest, Validating End-to-End Request-Response Cycles; Database Mocking - Mocking Database Queries (Mongoose, TypeORM), Testing CRUD Operations Without Affecting Production Databases; Advanced Assertions with Chai - Using Chai Assertions (expect, should, assert), Writing Tests with Chai Plugins (e.g., chai-http, chai-as-promised), Validating Nested Objects and Arrays; Testing Asynchronous Code - Managing Timers and Delays with setTimeout and Sinon.js, Testing Race Conditions and Multiple Async Operations; Code Coverage with NYC - Installing and Configuring NYC for Mocha, Generating and Analyzing Code Coverage Reports, Improving Test Coverage for Node.js and Express.js Applications.

Module 4 - Mocha Intermediate Concepts : Advanced Mocking with Sinon.js - Mocking Classes and Dependencies, Stubbing External Libraries and APIs, Resetting and Restoring Mocks Between Tests; Performance Testing - Measuring Execution Time for Functions, Testing for Performance Bottlenecks in Express.js Routes; Testing RESTful APIs - Sending HTTP Requests with Supertest, Testing API Endpoints for Pagination, Sorting, and Filtering, Simulating Authentication and Authorization Scenarios; Error Handling in Depth - Testing Custom Error Handlers in Express.js, Validating Responses for Different Error Scenarios; Working with Mock Files - Simulating File Uploads and Downloads, Testing APIs with File Handling Logic.

Module 5 – Miscellaneous : Testing Express.js Applications with Databases - Setting Up and Tearing Down Test Databases, Writing Integration Tests for Database Queries, Testing Relationships and Transactions; Security Testing - Validating Authentication Mechanisms (JWT), Testing Role-Based Access Control (RBAC) in Middleware, Simulating SQL Injection and XSS Scenarios; CI/CD Integration - Automating Mocha Tests in CI/CD Pipelines (Jenkins), Running Tests in Dockerized Environments, Generating Test Reports in JSON/HTML Formats; Organizing Tests in Large Applications - Structuring Test Suites for Scalability, Grouping Tests by Feature or Module, Writing Modular and Reusable Test Cases; Best Practices - Writing Maintainable and Readable Tests, Avoiding Flaky Tests with Reliable Mocking, Ensuring Comprehensive Test Coverage.

DevOps

Module 1 - Linux Fundamentals for DevOps : Introduction to Linux - Basics of Linux and Its Role in DevOps, Common Linux Distributions (Ubuntu, CentOS), Setting Up a Linux Environment (Local and Cloud); Basic Linux Commands - File and Directory Management (ls, cd, mkdir, rm), Viewing and Editing Files (cat, less, nano, vim), Managing Processes (ps, kill, top, htop), Permissions and Ownership (chmod, chown, sudo), Networking Commands (ifconfig, ping, netstat); Package Management - Installing and Managing Packages (apt, yum), Updating and Upgrading Linux Systems; Shell Scripting Basics - Writing Simple Bash Scripts, Automating Routine Tasks with Shell Scripts, Setting Up Cron Jobs for Scheduled Tasks; Managing Services - Starting, Stopping, and Restarting Services (systemctl, service), Logs Management Using journalctl and /var/log; Securing Linux Servers - Setting Up Basic Firewalls (ufw), SSH Key Management for Secure Remote Access.

Module 2 - AWS for Hosting and Managing Express.js Applications : Introduction to AWS - Understanding Cloud Computing and AWS Basics, Setting Up an AWS Account and CLI; Amazon EC2 (Elastic Compute Cloud) - Launching EC2 Instances, Connecting to EC2 Instances Using SSH, Installing Node.js and npm on EC2 for Express.js Applications, Configuring Security Groups for HTTP and HTTPS Traffic, Monitoring and Scaling Instances; Amazon S3 (Simple Storage Service) - Setting Up S3 Buckets, Uploading and Managing Static Assets (e.g., Images, Files), Using S3 with Express.js for File Uploads and Storage, Setting Bucket Policies and Permissions; Amazon RDS (Relational Database Service) - Setting Up MySQL/PostgreSQL Databases with RDS, Connecting Express.js Applications to RDS, Managing Backups and Snapshots; Amazon DynamoDB - Setting Up DynamoDB for NoSQL Databases, Integrating DynamoDB with Express.js for Simple Use Cases; AWS Elastic Load Balancer (ELB) - Configuring Load Balancers for High Availability, Distributing Traffic to Multiple EC2 Instances.

Module 3 - Continuous Integration and Delivery with Jenkins : Introduction to Jenkins, Why Use It for Express.js Applications?, Installing Jenkins on Linux or AWS EC2, Understanding Jenkins Pipeline Basics, Setting Up CI/CD Pipelines - Creating a Simple Pipeline for an Express.js Project, Automating Build, Test, and Deployment Processes, Using Jenkins with Git for Version Control; Building and Testing Express.js Applications - Integrating Unit Tests with Jenkins, Automating npm install, npm test, and npm run build Commands; Deploying Applications - Deploying to AWS EC2 via Jenkins Pipelines, Automating Deployment to AWS S3 for Static Files; Monitoring and Notifications - Setting Up Email and Slack Notifications for Build Status, Managing Build Logs and Debugging Pipeline Failures.

Module 4 - Containerization with Docker : Introduction to Docker, Why Use It for Express.js Applications, Installing Docker on Linux or AWS EC2, Understanding Docker Images and Containers; Working with Docker - Writing a Dockerfile for an Express.js Application, Building and Running Docker Containers, Exposing Ports and Environment Variables; Managing Docker Containers - Starting, Stopping, and Restarting Containers, Inspecting Logs and Resource Usage; Using docker-compose for Multi-Container Applications; Running Express.js with a Database (e.g., MongoDB, MySQL); Docker Hub - Pushing and Pulling Images from Docker Hub, Using Pre-Built Images for Databases (e.g., mysql, mongo); Basic Security - Managing Secrets with Docker, Avoiding Common Docker Security Pitfalls.

Module 5 - Orchestration with Kubernetes : Introduction to Kubernetes, Why Use It?, Key Kubernetes Concepts: Pods, Nodes, Deployments, and Services, Setting Up a Local Kubernetes Cluster (Minikube); Deploying Express.js Applications - Writing Kubernetes Deployment YAML Files, Exposing Applications with Services (NodePort, LoadBalancer), Using ConfigMaps and Secrets for Environment Variables; Scaling and Updates - Horizontal Pod Autoscaling (HPA) for Load Management, Rolling Updates and Rollbacks for Express.js Applications; Storage in Kubernetes - Using Persistent Volumes (PV) and Persistent Volume Claims (PVC), Integrating with AWS S3 or RDS for Persistent Data; Monitoring and Debugging - Monitoring Pod Logs and Application Metrics, Debugging Pods with kubectl Commands; Best Practices for Kubernetes - Using Namespaces for Resource Isolation, Managing Resource Limits and Requests, Securing Applications with Network Policies.

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.

Kafka, Zookeeper & RabbitMQ (Optional)

Module 1 - Introduction to Messaging Systems and Basics : Messaging Fundamentals - Overview of Messaging Systems, Differences Between Kafka, RabbitMQ, and Zookeeper, Use Cases in Node.js and Express.js Applications, Event-Driven Architectures and Microservices Integration, Key Concepts - Producers, Consumers, and Brokers, Topics, Queues, Partitions, and Exchanges; Installing Node.js Client Libraries: - kafkajs for Kafka, amqplib for RabbitMQ; Setting Up Development Environments:- Installing Apache Kafka and Zookeeper Locally or Using Docker, Installing RabbitMQ Locally or Using Docker; Building the First Application - Writing Simple Producers and Consumers in Kafka, Sending and Receiving Messages with RabbitMQ, Connecting Express.js with Kafka and RabbitMQ.

Module 2 - Kafka Essentials : Kafka Architecture - Brokers, Topics, Partitions, and Replication, Understanding Offset Management; Producers and Consumers - Creating Producers to Publish Events, Creating Consumers to Process Events; Kafka Topics - Creating and Managing Topics, Configuring Topic Properties (e.g., Partitions, Replication Factor); Message Handling and Processing - Producing Messages with kafkajs, Consuming Messages with Consumer Groups, Understanding and Managing Offsets, Using Kafka Streams for Real-Time Data Processing; Error Handling and Logging - Handling Consumer Failures, Retrying Failed Messages, Logging Kafka Events for Debugging.

Module 3 - Zookeeper Essentials : Introduction to Zookeeper - Role of Zookeeper in Kafka, Understanding Distributed Coordination, Use Cases of Zookeeper in Node.js Applications; Core Zookeeper Concepts - Zookeeper Architecture, Znodes and Hierarchical Namespace, Sessions and Watches, Configuring Zookeeper for Kafka, Setting Up and Managing a Zookeeper Ensemble; Using Zookeeper with Node.js - Connecting Node.js Applications to Zookeeper, Implementing Leader Election, Service Discovery Using Zookeeper; Monitoring and Security - Monitoring Zookeeper Health, Securing Zookeeper with Authentication and ACLs.

Module 4 - RabbitMQ Essentials : RabbitMQ Architecture - Queues, Exchanges, and Routing Keys, Virtual Hosts and Permissions; Types of Exchanges - Direct, Fanout, Topic, and Headers; Message Acknowledgments - Manual and Automatic Acknowledgments, Dead Letter Queues (DLQ); Producing and Consuming Messages - Using amqplib to Create Producers and Consumers, Publishing Messages to Exchanges, Subscribing to Queues, Managing Durable Queues and Persistent Messages ; Advanced Features - Message Prioritization and TTL, Delayed Messages with RabbitMQ Plugins, Implementing RPC (Remote Procedure Call) with RabbitMQ; Error Handling and Retry Mechanisms - Handling Failed Messages, Configuring Dead Letter Exchanges (DLX), Implementing Retry Logic for Consumers.

Module 5 - Integration, Monitoring, and Best Practices : Integration with Node.js and Express.js - Event-Driven Architecture in Express.js, Using Kafka for Asynchronous Communication, Using RabbitMQ for Task Queues; Building Microservices with Messaging Systems - Sending and Processing Events Between Microservices, Using Messaging Middleware in Express.js Applications; Monitoring and Logging - Kafka Monitoring Tools, Using Kafka Manager and Kafka UI, Monitoring Kafka Metrics (Lag, Throughput); RabbitMQ Monitoring Tools - RabbitMQ Management Plugin, Monitoring Queues, Connections, and Channels; Centralized Logging - Integrating Logs with Tools Like ELK Stack and Prometheus; Performance Optimization; Kafka - Optimizing Topic Partitions, Balancing Consumer Groups; RabbitMQ - Managing Connection Pools, Scaling Queues and Consumers; Zookeeper - Managing Load in Zookeeper Clusters, Reducing Latency in Service Discovery; Deployment and Security; Securing Kafka - Enabling SSL and SASL Authentication, Configuring Access Control Lists (ACLs); Securing RabbitMQ - Enabling TLS for Secure Connections, Managing Users and Permissions; Deploying in Production - Setting Up Kafka and RabbitMQ Clusters, Running Zookeeper as a Distributed Ensemble, Using Docker and Kubernetes for Containerized Deployments; Best Practices - Designing Scalable Messaging Systems, Ensuring Data Consistency and Message Ordering, Avoiding Common Pitfalls in Messaging Architectures, Testing Messaging Systems in Node.js Applications.

How to Join the Training Program

At **Learn2Earn Labs Training Institute**, we believe in nurturing talent and shaping the future of aspiring backend developers. 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, Information Technology, related engineering streams or any other related degree programs).
- **Passion for Learning**: Candidates must demonstrate a strong zeal to become a successful backend or software developer.
- **Growth-Oriented Mindset**: We are looking for individuals who are eager to learn, adapt to new challenges, and continuously improve their skills.
- **Hard Work and Dedication**: This program is rigorous and requires commitment. Candidates must be ready to put in the effort to achieve their career goals.

How to Apply

1. Visit our website at **www.LearntoEarnLabs.com** and navigate to the **"Contact Us"** section.
2. Fill out the application form with your details or query.
3. Share a brief statement explaining your motivation for joining the program and your career aspirations.
4. Complete any additional steps, such as interviews or assessments or telephonic discussion, as requested by the institute through 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'll receive all the details regarding the program schedule, enrollment ID, batch ID, terms & 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 backend developer, **this program is designed for you**. 😊

Join us today and pave the way to a successful career in backend development!

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, 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

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 (MERN Stack) HTML, CSS, JavaScript, ES, Git, ReactJS, Node, Express, MySQL & MongoDB

Full Stack Engineer Java, JavaScript, ReactJS, Spring, SpringBoot Hibernate, Linux, AWS, Git, DevOps & Jira

Digital Marketing Social Media, Website Design, SEO, Google Ads, Freelancing & Many More

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 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