**MockAttempt**

**TECH ACADEMY • OFFICIAL COURSE SYLLABUS BROCHURE**

MockAttempt Academy Pvt. Ltd. • Live Architect Mentorship, AI Assessments & Client Production

**Admissions & Support:** support@mockattempt.com
**Phone / WhatsApp:** +91 95400 07917
**Course Code:** FSWD • **Updated:** 05 Nov 2026
https://mockattempt.com/courses/full-stack-mern-typescript-cloud-engineering

**FSWD • FULL STACK DEVELOPMENT** **BEGINNER TO PRODUCTION ARCHITECT**

**100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER**

# Full Stack MERN & TypeScript Cloud Engineering

Go from core JavaScript runtime mechanics to deploying distributed React + Node.js microservices with timed Mock Tests, Voice AI Interviews, and Module Capstones.

**Duration:** 18–20 Weeks • Live + Hands-On Projects (232 Hrs) • **Modules:** 14 • **Live Topics:** 42 • **Subtopics:** 336+ • **Labs & Tasks:** 82 • **Mocks / AI Vivas:** 84 / 84 • **Projects:** 14 Module + 1 Live Client Prod

**All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹5,999** (Net Tuition ₹5,084 + Included 18% GST ₹915) • **Split Option:** ₹1,999 Lock + 2 × ₹2,000 • **Effective ROI:** ~₹26/Guided Hr (~₹143/Live Topic)

**Faculty:** 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 19–21 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 21–23 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (TypeScript 5 & JavaScript ES6+): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

**Core Technologies & Engineering Stack:** TypeScript • React 19 • Node.js • Express • MongoDB • Redis • Docker

## 1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

**End-to-End Engineering Outcomes:**

1. Understand V8 Event Loop, closures, async concurrency, and strict TypeScript generics
2. Architect scalable RESTful APIs with JWT rotation, rate limiting, and RBAC security
3. Build high-FPS React 19 interfaces with state machines, caching, and optimistic UI
4. Clear 42 Topic-Wise Timed Mock Exams and Voice AI Technical Interviews
5. Cohesive monorepo/workspace architecture sharing validated contracts between React 19 frontend and Node/Express APIs.
6. Crack LeetCode and HackerRank problems in TypeScript 5 & JavaScript ES6+ with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. Production authentication & multi-role RBAC using HttpOnly refresh tokens, CSRF/CORS protection, and secure middleware.
8. Indexed MongoDB document schemas, Aggregation Pipelines, Redis rate-limiting/caching, and S3 media upload flows.

**Eligibility & Prerequisites:** No prior Full Stack MERN & TypeScript Cloud Engineering industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Basic computer & internet familiarity on a Windows, macOS, or Linux laptop (8 GB+ RAM recommended for smooth live coding labs). • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews. • Dedication to write original code in your GitHub portfolio and complete the mentor-guided Live Client Production Sprint.

## 2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (14 MODULES • 42 LIVE TOPICS • 336+ SUBTOPICS)

**Progressive Zero-to-Expert Mastery Stages:** Modules 1–3: Stage 1: Day-1 Foundations & Core Mental Models (6 Topics / 48 Subtopics) • Modules 4–7: Stage 2: Applied Engineering, Data & API Workflows (14 Topics / 112 Subtopics) • Modules 8–11: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 12–14: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (10 Topics / 80 Subtopics)

**MODULE 1 OF 14 • STAGE 1 • 24 LIVE + LAB HOURS • 1 TOPICS (8 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

### JavaScript Runtime Internals & Strict TypeScript

Deep dive into V8 call stack, microtask queues, closures, and production TypeScript type safety.

**Objectives:** ✓ V8 Event Loop, Microtasks & Async Concurrency — How call stack, Web/Libuv APIs, Promise microtask queue, and macrotask timers execute. • ✓ Build, verify, and ship Module Capstone: Type-Safe Async Task Queue & Retry Engine.

**1.1 V8 Event Loop, Microtasks & Async Concurrency** [55 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** 1. Call Stack • 2. Microtask Queue • 3. Macrotask Queue • Understanding the Single • Execution Priority Order • JavaScript executes synchronous frames on a single Call Stack while delegating I/O, timers, and network sockets to.... • Reproduce microtask queue starvation in a 20 • Measure timer drift using performance.now()

**Purpose:** To understand and apply V8 Event Loop, Microtasks & Async Concurrency in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: How call stack, Web/L...

**What Topic Is Next → Module 1 Capstone (Type-Safe Async Task Queue & Retry Engine) → Next Topic 2.1: JWT Authentication, Refresh Rotation & OWASP API Security**

**Module 1 Hands-On Capstone Project (100 Marks):** Type-Safe Async Task Queue & Retry Engine — Design and implement a zero-dependency TypeScript task scheduler supporting concurrency limits, exponential backoff retries, cancellation via AbortController, and event hooks.

**MODULE 2 OF 14 • STAGE 1 • 32 LIVE + LAB HOURS • 1 TOPICS (8 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

### Node.js, Express & Secure REST API Architecture

Middleware pipelines, JWT refresh token rotation, MongoDB indexing, and Redis caching.

**Objectives:** ✓ JWT Authentication, Refresh Rotation & OWASP API Security — Stateless vs stateful sessions, token revocation, CSRF/XSS defenses, and rate limiting. • ✓ Build, verify, and ship Module Capstone: Multi-Tenant RBAC API Gateway with Redis Rate Limiting.

**2.1 JWT Authentication, Refresh Rotation & OWASP API Security** [60 mins • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Production API Authentication • Security Hygiene & Data Safety Rules for JWT Authentication, Refresh Rotation & OWASP API Security • Performance Optimization, Complexity & Latency Impact of JWT Authentication, Refresh Rotation & OWASP API Security • Live Debugging, Stack • JWT Authentication • Refresh Rotation • OWASP API Security • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Internals, Complexity & Latency Optimization for JWT Authentication

**Purpose:** To understand and apply JWT Authentication, Refresh Rotation & OWASP API Security in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Stateless...

**What Topic Is Next → Module 2 Capstone (Multi-Tenant RBAC API Gateway with Redis Rate Limiting) → Next Topic 3.1: Semantic Document Landmarks & WCAG Screen-Reader Forms**

**Module 2 Hands-On Capstone Project (100 Marks):** Multi-Tenant RBAC API Gateway with Redis Rate Limiting — Build an Express + TypeScript + MongoDB API featuring Argon2 password hashing, refresh token rotation, role-based access policies, and sliding-window rate limiting.

**MODULE 3 OF 14 • STAGE 1 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

### Semantic HTML5, Modern CSS3 Grid/Flexbox & Accessible Design Systems

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Semantic HTML5, Modern CSS3 Grid/Flexbox & Accessible Design Systems with hands-on labs and capstone build.

**Objectives:** ✓ Semantic Document Landmarks & WCAG Screen-Reader Forms — Purpose: Structure accessible web interfaces using semantic HTML5 elements and ARIA states. • ✓ CSS Flexbox, 2D Grid Dashboard Layouts & Container Queries — Purpose: Build fluid responsive SaaS dashboards that adapt cleanly from 320px mobile to 4K monitors. • ✓ Core Web Vitals (LCP, CLS, INP) & Critical Asset Optimization — Purpose: Eliminate layout jumps and slow initial paint on customer-facing web apps. • ✓ Utility-First Tailwind CSS Architecture & Reusable Variant Tokens — Purpose: Build maintainable responsive UI components rapidly without stylesheet bloat.

**3.1 Semantic Document Landmarks & WCAG Screen-Reader Forms** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Landmarks (<header>, <main>, <nav>, <section>) • Accessible Form Labels, Fieldsets & Validation States • Keyboard Focus Traps & Skip-Navigation Patterns • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Semantic Document Landmarks

**Purpose:** To understand and apply Semantic Document Landmarks & WCAG Screen-Reader Forms in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Structure ac...

**What Topic Is Next → Next Topic 3.2: CSS Flexbox, 2D Grid Dashboard Layouts & Container Queries**

**3.2 CSS Flexbox, 2D Grid Dashboard Layouts & Container Queries** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** 1D Flexbox Alignment vs 2D CSS Grid Template Areas • CSS Custom Properties (Design Tokens) & Dark Mode Themes • Container Queries (@container) & Fluid clamp() Typography • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for CSS Flexbox

**Purpose:** To understand and apply CSS Flexbox, 2D Grid Dashboard Layouts & Container Queries in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Build fl...

**What Topic Is Next → Next Topic 3.3: Core Web Vitals (LCP, CLS, INP) & Critical Asset Optimization**

### 3.3 Core Web Vitals (LCP, CLS, INP) & Critical Asset Optimization [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Eliminating Cumulative Layout Shift (CLS) with Aspect Ratios • Optimizing Largest Contentful Paint (LCP) & Responsive WebP/AVIF • Interaction to Next Paint (INP) & Non-Blocking UI Updates • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Core Web Vitals (LCP, CLS, INP)

**Purpose:** To understand and apply Core Web Vitals (LCP, CLS, INP) & Critical Asset Optimization in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Elimi...

**What Topic Is Next → Next Topic 3.4: Utility-First Tailwind CSS Architecture & Reusable Variant Tokens**

### 3.4 Utility-First Tailwind CSS Architecture & Reusable Variant Tokens [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Composable Utility Variants (Size, Tone & Disabled States) • Responsive Breakpoints (sm/md/lg/xl) & Mobile-First Ergonomics • Purging Unused Styles & Sub-20KB Gzipped Production Bundles • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Utility-First Tailwind CSS Architecture

**Purpose:** To understand and apply Utility-First Tailwind CSS Architecture & Reusable Variant Tokens in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: B...

**What Topic Is Next → Module 3 Capstone (Responsive Multi-Theme SaaS Design System & Accessible Shell) → Next Topic 4.1: Discriminated Unions, Exhaustive Type Narrowing & Type Guards**

**Module 3 Hands-On Capstone Project (100 Marks):** Responsive Multi-Theme SaaS Design System & Accessible Shell — Design, implement, test, and document 'Responsive Multi-Theme SaaS Design System & Accessible Shell' applying every production concept from Module 3 (Semantic HTML5, Modern CSS3 Grid/Flexbox...

**MODULE 4 OF 14 • STAGE 2 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Strict TypeScript Engineering: Generics, Narrowing & Runtime Contracts

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Strict TypeScript Engineering: Generics, Narrowing & Runtime Contracts with hands-on labs and capstone build.

**Objectives:** ✓ Discriminated Unions, Exhaustive Type Narrowing & Type Guards — Purpose: Eliminate undefined/null crashes across frontend and backend codebases at compile time. • ✓ Generic Constraints, Mapped Utility Types & Template Literal Types — Purpose: Write reusable, DRY TypeScript data layers without resorting to unsafe any casts. • ✓ Runtime Boundary Validation with Zod & Shared Monorepo Contracts — Purpose: Bridge compile-time TypeScript types with live runtime request validation. • ✓ Strict TSConfig Compiler Flags, ESM Bundling & Path Aliases — Purpose: Configure enterprise-grade TypeScript build pipelines for Node.js and Vite.

### 4.1 Discriminated Unions, Exhaustive Type Narrowing & Type Guards [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Discriminated State Unions (loading | success | error) • Custom User-Defined Type Predicates (val is User) • Exhaustiveness Checking with the never Type • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Discriminated Unions

**Purpose:** To understand and apply Discriminated Unions, Exhaustive Type Narrowing & Type Guards in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Elimi...

**What Topic Is Next → Next Topic 4.2: Generic Constraints, Mapped Utility Types & Template Literal Types**

### 4.2 Generic Constraints, Mapped Utility Types & Template Literal Types [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Generic Constraints (<T extends { id: string }>) • Built-In & Custom Mapped Types (Pick, Omit, Partial, Readonly) • Template Literal Event & Route Types • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Generic Constraints

**Purpose:** To understand and apply Generic Constraints, Mapped Utility Types & Template Literal Types in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 4.3: Runtime Boundary Validation with Zod & Shared Monorepo Contracts**

### 4.3 Runtime Boundary Validation with Zod & Shared Monorepo Contracts [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Zod Schema Parsing (.safeParse) & Inferred z.infer<typeof Schema> • Environment Variable Validation at Server Startup • Typed Error Envelopes & Problem-Details Responses • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Runtime Boundary Validation with Zod

**Purpose:** To understand and apply Runtime Boundary Validation with Zod & Shared Monorepo Contracts in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Br...

**What Topic Is Next → Next Topic 4.4: Strict TSConfig Compiler Flags, ESM Bundling & Path Aliases**

**4.4 Strict TSConfig Compiler Flags, ESM Bundling & Path Aliases** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** strictNullChecks, noUncheckedIndexedAccess & exactOptionalPropertyTypes • ECMAScript Modules (ESM) vs CommonJS Interop • Incremental Typechecking & Project References • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Strict TSConfig Compiler Flags

**Purpose:** To understand and apply Strict TSConfig Compiler Flags, ESM Bundling & Path Aliases in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Configu...

**What Topic Is Next → Module 4 Capstone (End-to-End Type-Safe API SDK & Schema Validator Library) → Next Topic 5.1: React Fiber Reconciliation, Keys & Declarative State Mental Model**

**Module 4 Hands-On Capstone Project (100 Marks):** End-to-End Type-Safe API SDK & Schema Validator Library — Design, implement, test, and document 'End-to-End Type-Safe API SDK & Schema Validator Library' applying every production concept from Module 4 (Strict TypeScript Engineering: Generics, Narr...

**MODULE 5 OF 14 • STAGE 2 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## React 19 Component Architecture, Reconciliation & Custom Hooks

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering React 19 Component Architecture, Reconciliation & Custom Hooks with hands-on labs and capstone build.

**Objectives:** ✓ React Fiber Reconciliation, Keys & Declarative State Mental Model — Purpose: Understand how React updates the DOM efficiently and avoid subtle state reset bugs. • ✓ Lifecycle Synchronization with useEffect, Cleanup & AbortController — Purpose: Fetch remote data, subscribe to sockets, and clean up timers without memory leaks. • ✓ Error Boundaries, Suspense Code-Splitting & Render Optimization — Purpose: Keep large React SPAs fast on slow networks and resilient when individual widgets fail. • ✓ Compound Components, Headless Portals & Accessible Modals — Purpose: Architect reusable enterprise component libraries used by multiple product squads.

**5.1 React Fiber Reconciliation, Keys & Declarative State Mental Model** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Virtual DOM Diffing & Stable List Keys • Controlled vs Uncontrolled Inputs & Ref Forwarding • Pure Render Functions & Idempotent UI Components • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for React Fiber Reconciliation

**Purpose:** To understand and apply React Fiber Reconciliation, Keys & Declarative State Mental Model in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: U...

**What Topic Is Next → Next Topic 5.2: Lifecycle Synchronization with useEffect, Cleanup & AbortController**

**5.2 Lifecycle Synchronization with useEffect, Cleanup & AbortController** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Effect Dependency Arrays & Stale Closure Prevention • Cancelling In-Flight Requests with AbortController • Extracting Domain Logic into Testable Custom Hooks • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Lifecycle Synchronization with useEffect

**Purpose:** To understand and apply Lifecycle Synchronization with useEffect, Cleanup & AbortController in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 5.3: Error Boundaries, Suspense Code-Splitting & Render Optimization**

**5.3 Error Boundaries, Suspense Code-Splitting & Render Optimization** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Route-Level Lazy Loading (React.lazy + Suspense) • Fault Isolation with Granular Error Boundaries • Profiling Re-Renders, useMemo, useCallback & List Virtualization • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Error Boundaries

**Purpose:** To understand and apply Error Boundaries, Suspense Code-Splitting & Render Optimization in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Kee...

**What Topic Is Next → Next Topic 5.4: Compound Components, Headless Portals & Accessible Modals**

**5.4 Compound Components, Headless Portals & Accessible Modals** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Compound Component Pattern (Tabs, Accordion, Dropdown) • React Portals for Modals, Drawers & Toast Notifications • Optimistic UI Transitions & Form Pending Indicators • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Compound Components

**Purpose:** To understand and apply Compound Components, Headless Portals & Accessible Modals in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Architect...

**What Topic Is Next → Module 5 Capstone (Modular Multi-Step SaaS Onboarding & Dynamic Form Builder) → Next Topic 6.1: Client State vs Server Cache Architecture with TanStack Query**

**Module 5 Hands-On Capstone Project (100 Marks):** Modular Multi-Step SaaS Onboarding & Dynamic Form Builder — Design, implement, test, and document 'Modular Multi-Step SaaS Onboarding & Dynamic Form Builder' applying every production concept from Module 5 (React 19 Component Architecture, Reconcilia...

**MODULE 6 OF 14 • STAGE 2 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Enterprise Frontend State: Redux Toolkit, TanStack Query & Protected Routing

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Enterprise Frontend State: Redux Toolkit, TanStack Query & Protected Routing with hands-on labs and capstone build.

**Objectives:** ✓ Client State vs Server Cache Architecture with TanStack Query — Separate ephemeral UI toggles from cached backend data with automatic background revalidation. • ✓ Predictable Global Store Design with Redux Toolkit & Selectors — Manage complex cross-screen client workflows like multi-item carts and studio editors. • ✓ Protected Role-Based Routing, URL Search State & Session Refresh — Synchronize filters with shareable URLs and handle silent JWT token renewal seamlessly. • ✓ Build, verify, and ship Module Capstone: Real-Time Operations Console with Optimistic Caching & RBAC Routes.

### 6.1 Client State vs Server Cache Architecture with TanStack Query [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Query Keys, Stale-While-Revalidate & Cache Invalidation • Optimistic Updates & Automatic Rollback on Mutation Failure • Cursor Infinite Scroll & Prefetched Pagination • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Client State vs Server Cache Architecture with TanSt...

**Purpose:** To understand and apply Client State vs Server Cache Architecture with TanStack Query in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Separ...

**What Topic Is Next → Next Topic 6.2: Predictable Global Store Design with Redux Toolkit & Selectors**

### 6.2 Predictable Global Store Design with Redux Toolkit & Selectors [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** createSlice, Immer Immutable Updates & Normalized Entity Adapters • Memoized Reselect Selectors • Middleware & DevTools Time-Travel Debugging • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Predictable Global Store Design with Redux Toolkit

**Purpose:** To understand and apply Predictable Global Store Design with Redux Toolkit & Selectors in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Mana...

**What Topic Is Next → Next Topic 6.3: Protected Role-Based Routing, URL Search State & Session Refresh**

### 6.3 Protected Role-Based Routing, URL Search State & Session Refresh [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Role-Guarded Route Wrappers & Deep-Link Redirects • URL Search Params as Shareable Filter State • Axios/Fetch Interceptor Queue for Silent Token Refresh • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Protected Role-Based Routing

**Purpose:** To understand and apply Protected Role-Based Routing, URL Search State & Session Refresh in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Sy...

**What Topic Is Next → Module 6 Capstone (Real-Time Operations Console with Optimistic Caching & RBAC Routes) → Next Topic 7.1: Node.js Readable/Writable/Transform Streams & Backpressure**

**Module 6 Hands-On Capstone Project (100 Marks):** Real-Time Operations Console with Optimistic Caching & RBAC Routes — Design, implement, test, and document 'Real-Time Operations Console with Optimistic Caching & RBAC Routes' applying every production concept from Module 6 (Enterprise Frontend State: Redux T...

**MODULE 7 OF 14 • STAGE 2 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Node.js Runtime Internals, Streams, Worker Threads & File Pipelines

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Node.js Runtime Internals, Streams, Worker Threads & File Pipelines with hands-on labs and capstone build.

**Objectives:** ✓ Node.js Readable/Writable/Transform Streams & Backpressure — Process multi-gigabyte uploads and exports in constant RAM without crashing containers. • ✓ CPU Offloading with Worker Threads, Child Processes & Cluster Mode — Keep the Node.js event loop responsive during PDF generation, hashing, and report crunching. • ✓ Structured Pino JSON Logging, AsyncLocalStorage & Request Tracing — Trace every log line and DB query belonging to a single customer request across async calls. • ✓ Build, verify, and ship Module Capstone: High-Throughput CSV/Log Streaming Ingestion & Worker Pool Engine.

### 7.1 Node.js Readable/Writable/Transform Streams & Backpressure [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** stream.pipeline() & Automatic File Descriptor Cleanup • Handling Backpressure (.write() false & drain Events) • Streaming NDJSON / CSV Exports Directly to HTTP Responses • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary

Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Node.js Readable/Writable/Transform Streams

**Purpose:** To understand and apply Node.js Readable/Writable/Transform Streams & Backpressure in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Process...

**What Topic Is Next → Next Topic 7.2: CPU Offloading with Worker Threads, Child Processes & Cluster Mode**

**7.2 CPU Offloading with Worker Threads, Child Processes & Cluster Mode** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Worker Thread Pools & SharedArrayBuffer / MessagePort • Graceful Shutdown on SIGTERM / SIGINT Signals • Buffers, Binary Encodings & Cryptographic HMAC Digests • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for CPU Offloading with Worker Threads

**Purpose:** To understand and apply CPU Offloading with Worker Threads, Child Processes & Cluster Mode in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 7.3: Structured Pino JSON Logging, AsyncLocalStorage & Request Tracing**

**7.3 Structured Pino JSON Logging, AsyncLocalStorage & Request Tracing** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Correlation Request IDs via AsyncLocalStorage • Redacting PII & Secrets from Structured Logs • Uncaught Rejection Policies & Health Check Endpoints • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Structured Pino JSON Logging

**Purpose:** To understand and apply Structured Pino JSON Logging, AsyncLocalStorage & Request Tracing in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: T...

**What Topic Is Next → Module 7 Capstone (High-Throughput CSV/Log Streaming Ingestion & Worker Pool Engine) → Next Topic 8.1: Layered Controller–Service–Repository Architecture & Domain Errors**

**Module 7 Hands-On Capstone Project (100 Marks):** High-Throughput CSV/Log Streaming Ingestion & Worker Pool Engine — Design, implement, test, and document 'High-Throughput CSV/Log Streaming Ingestion & Worker Pool Engine' applying every production concept from Module 7 (Node.js Runtime Internals, Streams,...

**MODULE 8 OF 14 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Production Express/REST API Engineering, Security Hardening & Idempotency

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Production Express/REST API Engineering, Security Hardening & Idempotency with hands-on labs and capstone build.

**Objectives:** ✓ Layered Controller–Service–Repository Architecture & Domain Errors — Organize Express backends so business rules stay decoupled from HTTP transport details. • ✓ OWASP API Security: Helmet, CORS, CSRF, XSS & Injection Defense — Harden Node.js APIs against the exact attack vectors exploited in real security breaches. • ✓ Idempotency Keys, Cursor Pagination & OpenAPI 3.1 Contracts — Prevent double-charging payments and slow offset queries on multi-million-row endpoints. • ✓ Build, verify, and ship Module Capstone: Hardened B2B Billing & Webhook API with Idempotency Keys.

**8.1 Layered Controller–Service–Repository Architecture & Domain Errors** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Thin Controllers, Pure Domain Services & Data Repositories • Centralized Error-Handling Middleware & RFC 7807 Problem JSON • API Versioning (/api/v1) & Backward-Compatible Deprecation • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Layered Controller–Service–Repository Architecture

**Purpose:** To understand and apply Layered Controller–Service–Repository Architecture & Domain Errors in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 8.2: OWASP API Security: Helmet, CORS, CSRF, XSS & Injection Defense**

**8.2 OWASP API Security: Helmet, CORS, CSRF, XSS & Injection Defense** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Strict CORS Origin Allowlists & Content-Security-Policy Headers • NoSQL/SQL Parameterization & Operator Sanitization ($gt, $ne) • Multipart Upload Validation, MIME Sniffing & Signed S3 URLs • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for OWASP API Security

**Purpose:** To understand and apply OWASP API Security: Helmet, CORS, CSRF, XSS & Injection Defense in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Har...

**What Topic Is Next → Next Topic 8.3: Idempotency Keys, Cursor Pagination & OpenAPI 3.1 Contracts**

**8.3 Idempotency Keys, Cursor Pagination & OpenAPI 3.1 Contracts** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Idempotency-Key Header Storage & Atomic Replay Protection • Keyset / Cursor Pagination vs Slow OFFSET Queries • Automated OpenAPI / Swagger Specification Generation • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Idempotency Keys

**Purpose:** To understand and apply Idempotency Keys, Cursor Pagination & OpenAPI 3.1 Contracts in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Prevent...

**What Topic Is Next → Module 8 Capstone (Hardened B2B Billing & Webhook API with Idempotency Keys) → Next Topic 9.1: Schema Design Patterns: Embedding vs Referencing & Polymorphic Docs**

**Module 8 Hands-On Capstone Project (100 Marks):** Hardened B2B Billing & Webhook API with Idempotency Keys — Design, implement, test, and document 'Hardened B2B Billing & Webhook API with Idempotency Keys' applying every production concept from Module 8 (Production Express/REST API Engineering, Sec...

**MODULE 9 OF 14 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## MongoDB Data Modeling, Aggregation Pipelines, Indexing & Transactions

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering MongoDB Data Modeling, Aggregation Pipelines, Indexing & Transactions with hands-on labs and capstone build.

**Objectives:** ✓ Schema Design Patterns: Embedding vs Referencing & Polymorphic Docs — Model document schemas around real application read/write access patterns. • ✓ Compound ESR Indexing, Explain Plans & Covered Queries — Cut slow 2-second collection scans down to single-digit millisecond B-Tree lookups. • ✓ Multi-Stage Aggregation Pipelines & Multi-Document ACID Transactions — Generate live revenue analytics inside MongoDB and guarantee atomic wallet transfers. • ✓ Build, verify, and ship Module Capstone: Multi-Tenant Analytics & Inventory Engine on MongoDB Replica Sets.

### 9.1 Schema Design Patterns: Embedding vs Referencing & Polymorphic Docs

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Bounded Subdocument Embedding vs ObjectId Referencing • Bucket & Computed Summary Patterns for Time-Series Data • Mongoose Schema Validation, Virtuals & Optimistic Versioning • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Schema Design Patterns

**Purpose:** To understand and apply Schema Design Patterns: Embedding vs Referencing & Polymorphic Docs in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 9.2: Compound ESR Indexing, Explain Plans & Covered Queries**

### 9.2 Compound ESR Indexing, Explain Plans & Covered Queries

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Equality–Sort–Range (ESR) Compound Index Rule • Reading .explain(\"executionStats\") (totalDocsExamined vs nReturned) • TTL Expiring Indexes & Partial Unique Indexes • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Compound ESR Indexing

**Purpose:** To understand and apply Compound ESR Indexing, Explain Plans & Covered Queries in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Cut slow 2-s...

**What Topic Is Next → Next Topic 9.3: Multi-Stage Aggregation Pipelines & Multi-Document ACID Transactions**

### 9.3 Multi-Stage Aggregation Pipelines & Multi-Document ACID Transactions

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** $match, $lookup, $unwind, $group & $facet Pipelines • Replica Set Sessions & Multi-Document ACID Transactions • Read Concerns, Write Concerns (majority) & Change Streams • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Multi-Stage Aggregation Pipelines

**Purpose:** To understand and apply Multi-Stage Aggregation Pipelines & Multi-Document ACID Transactions in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically...

**What Topic Is Next → Module 9 Capstone (Multi-Tenant Analytics & Inventory Engine on MongoDB Replica Sets) → Next Topic 10.1: Cache-Aside, Write-Through, TTL Jitter & Stampede Protection**

**Module 9 Hands-On Capstone Project (100 Marks):** Multi-Tenant Analytics & Inventory Engine on MongoDB Replica Sets — Design, implement, test, and document 'Multi-Tenant Analytics & Inventory Engine on MongoDB Replica Sets' applying every production concept from Module 9 (MongoDB Data Modeling, Aggregation...

**MODULE 10 OF 14 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Redis Caching Strategies, Distributed Locks & BullMQ Background Jobs

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Redis Caching Strategies, Distributed Locks & BullMQ Background Jobs with hands-on labs and capstone build.

**Objectives:** ✓ Cache-Aside, Write-Through, TTL Jitter & Stampede Protection — Slash database load by 90% on read-heavy catalogs and user profile endpoints. • ✓ Atomic Lua Sliding-Window Rate Limiting & Redlock Concurrency — Protect login, OTP, and checkout endpoints against brute-force and race conditions. • ✓

Asynchronous Background Queues, Retries & Dead-Letter Handling with BullMQ — Move slow emails, PDF invoices, and webhooks out of user-facing HTTP requests. • ✓ Build, verify, and ship Module Capstone: Distributed Job Queue, Rate Limiter & Sub-5ms Cache Layer.

**10.1 Cache-Aside, Write-Through, TTL Jitter & Stampede Protection** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Cache-Aside Pattern & Targeted Tag/Key Invalidation • Randomized TTL Jitter & Single-Flight Mutex Locks • Redis Hashes, Sorted Sets (ZSET) & Leaderboards • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Cache-Aside

**Purpose:** To understand and apply Cache-Aside, Write-Through, TTL Jitter & Stampede Protection in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Slash...

**What Topic Is Next → Next Topic 10.2: Atomic Lua Sliding-Window Rate Limiting & Redlock Concurrency**

**10.2 Atomic Lua Sliding-Window Rate Limiting & Redlock Concurrency** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Atomic Redis Lua Scripts for Race-Free Counters • Distributed Mutex Locks (SET NX PX) for Inventory Booking • Token Blacklisting & Instant Multi-Device Session Revocation • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Atomic Lua Sliding-Window Rate Limiting

**Purpose:** To understand and apply Atomic Lua Sliding-Window Rate Limiting & Redlock Concurrency in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Prote...

**What Topic Is Next → Next Topic 10.3: Asynchronous Background Queues, Retries & Dead-Letter Handling with BullMQ**

**10.3 Asynchronous Background Queues, Retries & Dead-Letter Handling with BullMQ** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Producer–Worker Decoupling for Sub-50ms HTTP Responses • Exponential Backoff Retries & Dead-Letter Queues (DLQ) • Scheduled Cron Jobs & Idempotent Job Handlers • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Asynchronous Background Queues

**Purpose:** To understand and apply Asynchronous Background Queues, Retries & Dead-Letter Handling with BullMQ in real Full Stack MERN & TypeScript Cloud Engineering systems — specif...

**What Topic Is Next → Module 10 Capstone (Distributed Job Queue, Rate Limiter & Sub-5ms Cache Layer) → Next Topic 11.1: WebSockets vs Server-Sent Events (SSE) & Authenticated Handshakes**

**Module 10 Hands-On Capstone Project (100 Marks):** Distributed Job Queue, Rate Limiter & Sub-5ms Cache Layer — Design, implement, test, and document 'Distributed Job Queue, Rate Limiter & Sub-5ms Cache Layer' applying every production concept from Module 10 (Redis Caching Strategies, Distributed Lock...

**MODULE 11 OF 14 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Real-Time WebSockets, Event-Driven Notifications & Collaborative State

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Real-Time WebSockets, Event-Driven Notifications & Collaborative State with hands-on labs and capstone build.

**Objectives:** ✓ WebSockets vs Server-Sent Events (SSE) & Authenticated Handshakes — Choose the right real-time protocol and authenticate persistent connections securely. • ✓ Horizontal Socket Scaling with Redis Pub/Sub Adapters — Broadcast real-time events across multiple load-balanced Node.js servers. • ✓ Transactional Outbox & Reliable In-App Notification Center — Guarantee real-time events and persistent unread badges never drift out of sync. • ✓ Build, verify, and ship Module Capstone: Live Multi-User Kanban & Instant Notification Hub with Socket.IO.

**11.1 WebSockets vs Server-Sent Events (SSE) & Authenticated Handshakes** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Full-Duplex WebSockets vs Unidirectional HTTP SSE • JWT Handshake Verification & Per-Tenant Room Isolation • Heartbeats, Exponential Reconnection & Missed-Event Catchup • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for WebSockets vs Server-Sent Events (SSE)

**Purpose:** To understand and apply WebSockets vs Server-Sent Events (SSE) & Authenticated Handshakes in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: C...

**What Topic Is Next → Next Topic 11.2: Horizontal Socket Scaling with Redis Pub/Sub Adapters**

**11.2 Horizontal Socket Scaling with Redis Pub/Sub Adapters** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Sticky Load Balancing & Redis Pub/Sub Backplane • Presence Tracking (Online/Typing Indicators) with Expiring Keys • Backpressure & Throttling High-Frequency Cursor/Ticker Events • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Horizontal Socket Scaling with Redis Pub/Sub Adapter...

**Purpose:** To understand and apply Horizontal Socket Scaling with Redis Pub/Sub Adapters in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Broadcast rea...

**What Topic Is Next → Next Topic 11.3: Transactional Outbox & Reliable In-App Notification Center**

### 11.3 Transactional Outbox & Reliable In-App Notification Center [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Persisting Notifications Before Publishing Socket Events • Batch Mark-as-Read & Unread Counter Denormalization • Outbound Customer Webhooks with Signed Delivery Logs • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Transactional Outbox

**Purpose:** To understand and apply Transactional Outbox & Reliable In-App Notification Center in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Guarante...

**What Topic Is Next → Module 11 Capstone (Live Multi-User Kanban & Instant Notification Hub with Socket.IO) → Next Topic 12.1: Unit & Integration Testing with Vitest, React Testing Library & Supertest**

**Module 11 Hands-On Capstone Project (100 Marks):** Live Multi-User Kanban & Instant Notification Hub with Socket.IO — Design, implement, test, and document 'Live Multi-User Kanban & Instant Notification Hub with Socket.IO' applying every production concept from Module 11 (Real-Time WebSockets, Event-Driven...

**MODULE 12 OF 14 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Automated Testing (Vitest, Supertest, Playwright), Docker & Cloud CI/CD

Production-grade Full Stack MERN & TypeScript Cloud Engineering engineering module covering Automated Testing (Vitest, Supertest, Playwright), Docker & Cloud CI/CD with hands-on labs and capstone build.

**Objectives:** ✓ Unit & Integration Testing with Vitest, React Testing Library & Supertest — Purpose: Verify frontend components and Express APIs automatically on every Git pull request. • ✓ End-to-End Browser Automation with Playwright — Purpose: Simulate real customer signup, checkout, and dashboard journeys in headless Chromium. • ✓ Multi-Stage Dockerfiles, Compose Stacks & Non-Root Container Security — Purpose: Package React + Node.js apps into lean, reproducible Linux containers. • ✓ GitHub Actions CI/CD, Nginx TLS Reverse Proxy & Production Observability — Purpose: Automate linting, testing, container publishing, and zero-downtime HTTPS deployment.

### 12.1 Unit & Integration Testing with Vitest, React Testing Library & Supertest [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Testing User Behavior with React Testing Library (Not Implementation Details) • API Integration Tests Against Ephemeral MongoDB/Redis Instances • Mocking External Payment/Email Gateways Deterministically • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Unit

**Purpose:** To understand and apply Unit & Integration Testing with Vitest, React Testing Library & Supertest in real Full Stack MERN & TypeScript Cloud Engineering systems — specifi...

**What Topic Is Next → Next Topic 12.2: End-to-End Browser Automation with Playwright**

### 12.2 End-to-End Browser Automation with Playwright [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Role-Based Locators & Auto-Waiting Assertions • Authenticated StorageState Reuse Across Test Suites • Trace Viewer Screenshots & Network HAR Capture on Failure • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for End-to-End Browser Automation with Playwright

**Purpose:** To understand and apply End-to-End Browser Automation with Playwright in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Simulate real custome...

**What Topic Is Next → Next Topic 12.3: Multi-Stage Dockerfiles, Compose Stacks & Non-Root Container Security**

### 12.3 Multi-Stage Dockerfiles, Compose Stacks & Non-Root Container Security [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Multi-Stage Builds (Build Stage -> Alpine/Distroless Runtime) • Running Node as Non-Root User & Read-Only Filesystems • Docker Compose Local Parity (App + Mongo + Redis + Nginx) • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Multi-Stage Dockerfiles

**Purpose:** To understand and apply Multi-Stage Dockerfiles, Compose Stacks & Non-Root Container Security in real Full Stack MERN & TypeScript Cloud Engineering systems — specificall...

**What Topic Is Next → Next Topic 12.4: GitHub Actions CI/CD, Nginx TLS Reverse Proxy & Production Observability**

### 12.4 GitHub Actions CI/CD, Nginx TLS Reverse Proxy & Production Observability [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** GitHub Actions Matrix Pipelines (Typecheck -> Test -> Build -> Deploy) • Nginx Reverse Proxy, Let’s Encrypt HTTPS & Gzip/Brotli Compression • Error Tracking (Sentry), Uptime Alerts & Rollback Runbooks • Under-the-Hood Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle • Boundary Validation, HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards & Failure Recovery • Automated Vitest, Supertest & Playwright E2E Verification & Senior Viva Trade-Offs • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys &

Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for GitHub Actions CI/CD

**Purpose:** To understand and apply GitHub Actions CI/CD, Nginx TLS Reverse Proxy & Production Observability in real Full Stack MERN & TypeScript Cloud Engineering systems — specific...

**What Topic Is Next → Module 12 Capstone (Containerized MERN Production Pipeline with Automated E2E & Zero-Downtime Release) → Next Topic 13.1: Tenant Context Isolation, Row-Level Guards & Scoped MongoDB Queries**

**Module 12 Hands-On Capstone Project (100 Marks):** Containerized MERN Production Pipeline with Automated E2E & Zero-Downtime Release — Design, implement, test, and document 'Containerized MERN Production Pipeline with Automated E2E & Zero-Downtime Release' applying every production concept from Module 12 (Automated Testing...

**MODULE 13 OF 14 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Multi-Tenant SaaS Security, Rate-Limiting & Real-Time WebSockets

Self-explanatory list-and-bullet production mastery module covering Multi-Tenant SaaS Security, Rate-Limiting & Real-Time WebSockets in Full Stack MERN & TypeScript Cloud Engineering, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

**Objectives:** ✓ Tenant Context Isolation, Row-Level Guards & Scoped MongoDB Queries — Purpose: Prevent cross-tenant data leaks in multi-company B2B SaaS platforms using scoped middleware and indexed tenant keys. • ✓ Socket.IO / WebSockets Authenticated Rooms, Heartbeats & Redis Pub/Sub Scaling — Purpose: Broadcast live collaborative workspace updates across multiple Node.js server instances without dropped events. • ✓ Idempotency Keys, Webhook Signature Verification & Audit Trails — Purpose: Guarantee billing webhooks and state-mutating API calls never double-charge or corrupt tenant records on retries. • ✓ Build, verify, and document 'Tenant-Isolated Real-Time Notification & Sliding-Window Limiter Service' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

### 13.1 Tenant Context Isolation, Row-Level Guards & Scoped MongoDB Queries

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Tenant Context Isolation, Row • Performance Optimization, Complexity & Latency Impact of Tenant Context Isolation, Row • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Tenant Context Isolation • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply Tenant Context Isolation, Row-Level Guards & Scoped MongoDB Queries in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Next Topic 13.2: Socket.IO / WebSockets Authenticated Rooms, Heartbeats & Redis Pub/Sub Scaling**

### 13.2 Socket.IO / WebSockets Authenticated Rooms, Heartbeats & Redis Pub/Sub Scaling

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Socket.IO / WebSockets Authenticated Rooms, Heartbeats & Redis Pub/Sub Sca... • Performance Optimization, Complexity & Latency Impact of Socket.IO / WebSockets Authenticated Rooms, Heartbeats & R... • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Socket.IO / WebSockets Authenticated Rooms • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply Socket.IO / WebSockets Authenticated Rooms, Heartbeats & Redis Pub/Sub Scaling in real Full Stack MERN & TypeScript Cloud Engineering systems — sp...

**What Topic Is Next → Next Topic 13.3: Idempotency Keys, Webhook Signature Verification & Audit Trails**

### 13.3 Idempotency Keys, Webhook Signature Verification & Audit Trails

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Idempotency Keys, Webhook Signature Verification & Audit Trails • Performance Optimization, Complexity & Latency Impact of Idempotency Keys, Webhook Signature Verification & Audit T... • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Idempotency Keys • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply Idempotency Keys, Webhook Signature Verification & Audit Trails in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: Gua...

**What Topic Is Next → Module 13 Capstone (Tenant-Isolated Real-Time Notification & Sliding-Window Limiter Service) → Next Topic 14.1: OpenTelemetry Distributed Tracing Across React Fetch, Express & MongoDB**

**Module 13 Hands-On Capstone Project (100 Marks):** Tenant-Isolated Real-Time Notification & Sliding-Window Limiter Service — Architect, implement, test, and document 'Tenant-Isolated Real-Time Notification & Sliding-Window Limiter Service' applying every bullet-wise engineering concept from Module 13 (Multi-Tenant...

**MODULE 14 OF 14 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)**

## Full Stack Observability, Load Testing & System Design Interview Readiness

Self-explanatory list-and-bullet production mastery module covering Full Stack Observability, Load Testing & System Design Interview Readiness in Full Stack MERN & TypeScript Cloud Engineering, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

**Objectives:** ✓ OpenTelemetry Distributed Tracing Across React Fetch, Express & MongoDB — Purpose: Pinpoint exact p95 latency bottlenecks from browser click to database query in live production. • ✓ k6 Load Benchmarking, Connection Pool Sizing & Graceful Shutdowns — Purpose: Verify zero dropped HTTP requests during rolling Docker deployments and sudden traffic spikes. • ✓ Full Stack MERN Architecture Trade-Offs & Live Production Debugging — Purpose:

Solve real product-company system design scenarios, memory leak diagnostics, and stale-cache bugs confidently. • ✓ Build, verify, and document 'OpenTelemetry Full-Stack Trace Dashboard & k6 Stress-Test Harness' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

**14.1 OpenTelemetry Distributed Tracing Across React Fetch, Express & MongoDB** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for OpenTelemetry Distributed Tracing Across React Fetch, Express & MongoDB • Performance Optimization, Complexity & Latency Impact of OpenTelemetry Distributed Tracing Across React Fetch, Expr... • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for OpenTelemetry Distributed Tracing Across React Fetch • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply OpenTelemetry Distributed Tracing Across React Fetch, Express & MongoDB in real Full Stack MERN & TypeScript Cloud Engineering systems — specifica...

**What Topic Is Next → Next Topic 14.2: k6 Load Benchmarking, Connection Pool Sizing & Graceful Shutdowns**

**14.2 k6 Load Benchmarking, Connection Pool Sizing & Graceful Shutdowns** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for k6 Load Benchmarking, Connection Pool Sizing & Graceful Shutdowns • Performance Optimization, Complexity & Latency Impact of k6 Load Benchmarking, Connection Pool Sizing & Graceful Sh... • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for k6 Load Benchmarking • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply k6 Load Benchmarking, Connection Pool Sizing & Graceful Shutdowns in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically: V...

**What Topic Is Next → Next Topic 14.3: Full Stack MERN Architecture Trade-Offs & Live Production Debugging**

**14.3 Full Stack MERN Architecture Trade-Offs & Live Production Debugging** [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

**Subtopics Covered:** Security Hygiene & Data Safety Rules for Full Stack MERN Architecture Trade • Performance Optimization, Complexity & Latency Impact of Full Stack MERN Architecture Trade • Live Debugging, Stack • Runnable Code Implementation (TYPESCRIPT) • Hands • Idiomatic FSWD Implementation: strict TypeScript contracts, non-blocking I/O, idempotency keys & Redis cachin... • Node.js 22 V8 + Libuv Event Loop, Streams & React/Next.js Full-Stack Execution & Memory Lifecycle for Full Stack MERN Architecture Trade-Offs • Production Hardening & Security: HttpOnly rotating JWTs, Zod boundary schemas, Helmet & OWASP BOLA guards

**Purpose:** To understand and apply Full Stack MERN Architecture Trade-Offs & Live Production Debugging in real Full Stack MERN & TypeScript Cloud Engineering systems — specifically:...

**What Topic Is Next → Module 14 Capstone → End-of-Course Live Client Project: CloudPulse Multi-Tenant B2B SaaS CRM & Automated Billing Portal**

**Module 14 Hands-On Capstone Project (100 Marks):** OpenTelemetry Full-Stack Trace Dashboard & k6 Stress-Test Harness — Architect, implement, test, and document 'OpenTelemetry Full-Stack Trace Dashboard & k6 Stress-Test Harness' applying every bullet-wise engineering concept from Module 14 (Full Stack Observa...

## 3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

**End-of-Course Live Client Production Project + Official Internship Confirmation Letter**

**Project:** CloudPulse Multi-Tenant B2B SaaS CRM & Automated Billing Portal • **Confirmed Role:** Full Stack MERN & Cloud Engineering Intern

**Domain & Production Infra:** Enterprise B2B SaaS & FinTech Client Delivery (Docker + GitHub Actions CI/CD + AWS Nginx HTTPS Production Cluster)

- Architected React 19 + TypeScript workspace dashboards with optimistic state updates and role-based access.
- Engineered Node.js + Express REST APIs with HttpOnly JWT refresh token rotation and sliding-window Redis rate limits.
- Optimized MongoDB compound indexes and aggregation pipelines for sub-50ms multi-tenant analytics.
- Implemented automated unit/integration test suites and zero-downtime Docker CI/CD deployment to live client production.

**Target Job Roles:** Full-Stack MERN & TypeScript Engineer, Product Software Engineer (Full Stack), Frontend + Backend Cloud Developer, SDE-1 / SDE-2 Full Stack

**Realistic Annual CTC Scope:** **Fresher / Entry-Level Full-Stack SDE (0–1 Yr):** ₹6.0 LPA – ₹12.0 LPA **Mid-Level Full-Stack Product Engineer (1–3 Yrs):** ₹12.0 LPA – ₹24.0 LPA **Senior Full-Stack / Cloud Lead Architect (3–6+ Yrs):** ₹24.0 LPA – ₹46.0+ LPA