Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: NODE • **Updated:** 01 Nov 2026
https://mockattempt.com/courses/node-js-training

NODE • BACKEND & CLOUD RUNTIME ENGINEERING **BEGINNER TO PRODUCTION NODE.JS BACKEND ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

Node js Training

Master production Node.js from V8 + Libuv Event Loop internals, Buffers, Backpressure Streams, FS & Crypto to Child Processes, Express.js REST APIs, MySQL2 Pools, MongoDB Aggregations & CI/CD — with runnable Node.js programs, 10×50 MCQ Mock Tests per topic, and Spoken Voice + Gemini AI Interviews at ₹3,999.

Duration: 16–18 Weeks • Live + Hands-On Projects (216 Hrs) • **Modules:** 16 • **Live Topics:** 53 • **Labs & Tasks:** 106 • **Mocks / AI Vivas:** 386 / 386 • **Projects:** 16 Module + 1 Live Client Prod

Fixed Non-Negotiable Tuition Fee: Base Fee **₹3,999** + 18% GST (₹720) = **Total Incl. GST: ₹4,719**

Live Batch Tracks: Weekday Live Online Batch (Mon – Fri) (Monday to Friday (5 Live Interactive Classes / Week) • 1.5 Hours / Day (60 Mins Live Architecture & Coding + 30 Mins Hands-On Lab) • 18–20 Weeks Fast-Track) | Weekend Intensive Live Online Batch (Sat – Sun) (Saturday & Sunday (Designed for Working Professionals & College Students) • 3.0 Hours / Day (2 Hrs Deep-Dive Live Coding + 1 Hr Guided Project Build) • 20–22 Weeks Weekend Track)

Core Technologies & Engineering Stack: Node.js 22 LTS • V8 & Libuv Event Loop • Streams & Buffers • Express.js REST APIs • node:crypto & TLS • MySQL2 & MongoDB • node:test & CI/CD

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Master V8 JIT compilation, Libuv non-blocking Event Loop phases, process.nextTick vs Promises, and race-condition prevention
2. Process multi-gigabyte payloads in constant RAM using Binary Buffers, Readable/Writable/Transform Streams, pipeline() backpressure, and Zlib
3. Engineer hardened Express.js REST APIs with modular routers, timing-safe HMAC/TLS security, Child Process scaling, MySQL2 pools, and MongoDB indexes
4. Clear 350 Timed Mock Exams (17,500 Node.js MCQs) and 350 Spoken Voice + Gemini AI Backend Technical Interviews
5. Non-blocking Event Loop architecture, Streams, Worker Threads, and clean Controller → Service → Repository layers.
6. Schema-validated REST APIs with centralized error middleware, idempotency keys, and consistent HTTP status contracts.
7. Defense-in-depth security: HttpOnly JWT rotation, RBAC middleware, Helmet headers, CORS policies, and rate limiting.
8. Connection-pooled SQL/NoSQL data access, webhook HMAC signature verification, and background queue processing.

Eligibility & Prerequisites: Working knowledge of core JavaScript (functions, objects, arrays, Promises, async/await) • Any laptop/PC running Node.js 20/22 LTS, npm, and VS Code • No prior Node js 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.

2. DETAILED MODULE & TOPIC-WISE SYLLABUS (16 MODULES • 53 LIVE TOPICS)

MODULE 1 OF 16 • 14 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: Node.js Runtime Architecture, V8 Engine, Libuv, NPM & ESM vs CommonJS

Understand why single-threaded event-driven Node.js handles tens of thousands of concurrent connections, V8 JIT compilation & heap spaces, npm SemVer lockfiles, and ES Modules (`import`) vs CommonJS (`require`).

Objectives: ✓ Explain how V8 + Libuv achieve high-concurrency I/O on a single JavaScript main thread • ✓ Inspect V8 heap statistics (`v8.getHeapStatistics()`) and garbage collection generations • ✓ Manage deterministic `package-lock.json` dependencies with `npm ci` and migrate cleanly between CommonJS and ES Modules

1.1 Node.js Architecture — Event-Driven Non-Blocking I/O, Single JS Thread & Libuv Overview [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Node.js Architecture — Event-Driven Non-Blocking I/O, Single JS Thread & Libuv Overview in real Node js systems — specifically: What Node.js is (C...

1.2 Google V8 JavaScript Engine — Ignition Bytecode, TurboFan JIT & Generational Heap GC [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Google V8 JavaScript Engine — Ignition Bytecode, TurboFan JIT & Generational Heap GC in real Node js systems — specifically: How V8 parses JS into...

1.3 NPM & Package Governance — `package.json`, SemVer (`^` vs `~`), `package-lock.json` & `npm ci` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply NPM & Package Governance — `package.json`, SemVer (`^` vs `~`), `package-lock.json` & `npm ci` in real Node js systems — specifically: Authoring `...

1.4 CommonJS (`require`/`module.exports`) vs ES Modules (`import`/`export`) in Node.js [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply CommonJS (`require`/`module.exports`) vs ES Modules (`import`/`export`) in Node.js in real Node js systems — specifically: Synchronous dynamic `re...

Module 1 Hands-On Capstone Project (100 Marks): Production ESM Node.js Workspace & V8 Heap Telemetry CLI — Build an ES Module (`"type": "module"`) Node.js CLI diagnostics tool that prints Node/V8 versions, live heap space breakdown via `node:v8`, dependency lockfile audit checks, and non-blocking...

MODULE 2 OF 16 • 12 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Node.js Globals, `process` Lifecycle, Signals, `node:os` Telemetry & Timers

Master `globalThis`, `process.env` validation, graceful `SIGTERM`/`SIGINT` shutdown hooks, `node:os` CPU/memory inspection, and Event Loop timers (`setImmediate`, `setTimeout`, `AbortSignal.timeout`).

Objectives: ✓ Avoid polluting `globalThis` and understand why Node has no `window`/`document` globals • ✓ Implement zero-dropped-request graceful shutdown on `SIGTERM`/`SIGINT` using `process.on` • ✓ Control timer phases (`setTimeout`, `setImmediate`, `timer.unref()`) without keeping CLI processes hanging

2.1 Node.js Global Scope — `globalThis`, `URL`, `fetch`, `AbortController`, `structuredClone` & Module Scope [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Node.js Global Scope — `globalThis`, `URL`, `fetch`, `AbortController`, `structuredClone` & Module Scope in real Node js systems — specifically: T...

2.2 The `node:process` Object — `process.env`, `argv`, `memoryUsage()`, Exit Codes & `SIGTERM` Shutdown [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply The `node:process` Object — `process.env`, `argv`, `memoryUsage()`, Exit Codes & `SIGTERM` Shutdown in real Node js systems — specifically: Readin...

2.3 Operating System Module (`node:os`) — `availableParallelism()`, Memory, Endianness & Host Metrics [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Operating System Module (`node:os`) — `availableParallelism()`, Memory, Endianness & Host Metrics in real Node js systems — specifically: Querying...

2.4 Node.js Timers (`node:timers`, `timers/promises`) — `setImmediate`, `setTimeout`, `unref()` & Cancellation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Node.js Timers (`node:timers`, `timers/promises`) — `setImmediate`, `setTimeout`, `unref()` & Cancellation in real Node js systems — specifically:...

Module 2 Hands-On Capstone Project (100 Marks): Zero-Downtime Graceful Shutdown Supervisor & Host Telemetry Agent — Build a background Node.js health daemon that validates required `process.env` keys at boot, reports `node:os` load averages & free memory every interval using `.unref()` timers, and drains...

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

Module 3: Event Loop Phases, Callbacks vs Promises, `EventEmitter` Pub/Sub & Concurrency Safety

Trace every Libuv Event Loop phase, convert error-first callbacks with `util.promisify`, build memory-safe `EventEmitter` buses, centralize async error recovery, and eliminate async race conditions.

Objectives: ✓ Refactor nested error-first callbacks `(err, data)` to clean `async/await` without Zalgo mixed sync/async hazards • ✓ Build decoupled architectures with `node:events` (`EventEmitter`, `events.once`) without listener leaks • ✓ Prevent TOCTOU and double-spend async race conditions across `await` suspension points

3.1 Error-First Callbacks `(err, result)`, Callback Hell, Zalgo Anti-Pattern & `util.promisify` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Error-First Callbacks `(err, result)`, Callback Hell, Zalgo Anti-Pattern & `util.promisify` in real Node js systems — specifically: Node's classic...

3.2 `node:events` & `EventEmitter` — Synchronous Dispatch, `once()`, `error` Events & Listener Leaks [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply `node:events` & `EventEmitter` — Synchronous Dispatch, `once()`, `error` Events & Listener Leaks in real Node js systems — specifically: Pub/Sub i...

3.3 Error Architecture in Node.js — Operational vs Programmer Errors, `Error.cause` & Unhandled Rejections [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Error Architecture in Node.js — Operational vs Programmer Errors, `Error.cause` & Unhandled Rejections in real Node js systems — specifically: Sys...

3.4 Race Conditions in Single-Threaded Node.js — Interleaved `await` Points, TOCTOU & Atomic Locks [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Race Conditions in Single-Threaded Node.js — Interleaved `await` Points, TOCTOU & Atomic Locks in real Node js systems — specifically: Why "single...

Module 3 Hands-On Capstone Project (100 Marks): Resilient EventEmitter Order Bus with Keyed Async Mutex & Dead-Letter Error Queue — Engineer an in-memory Node.js Order Processing Bus extending `EventEmitter` that deduplicates concurrent wallet debits per `userId` using a keyed async mutex, handles `'error'` events safely...

MODULE 4 OF 16 • 14 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Binary `Buffer`, Backpressure `Stream` Pipelines, Async `node:fs` & Safe `node:path`

Manipulate raw binary bytes safely with `Buffer`, process 10 GB files in 32 MB RAM using `stream/promises` `pipeline()`, master `node:fs/promises`, and block path-traversal attacks with `node:path`.

Objectives: ✓ Allocate zero-filled buffers safely (`Buffer.alloc` vs `allocUnsafe`) and convert UTF-8/Base64/Hex • ✓ Compose Readable, Transform, and Writable streams with `pipeline()` for automatic backpressure & error teardown • ✓ Prevent `../../etc/passwd` directory traversal using `path.resolve` containment checks

4.1 Node.js `Buffer` (`node:buffer`) — `Buffer.alloc` vs `allocUnsafe`, UTF-8/Base64/Hex & `Uint8Array` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Node.js `Buffer` (`node:buffer`) — `Buffer.alloc` vs `allocUnsafe`, UTF-8/Base64/Hex & `Uint8Array` in real Node js systems — specifically: Fixed-...

4.2 Node.js Streams (`node:stream`) — Readable, Writable, Transform, Backpressure & `pipeline()` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Node.js Streams (`node:stream`) — Readable, Writable, Transform, Backpressure & `pipeline()` in real Node js systems — specifically: Four stream t...

4.3 File System (`node:fs/promises`) — Non-Blocking Read/Write, `createReadStream`, Atomic Rename & `stat` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply File System (`node:fs/promises`) — Non-Blocking Read/Write, `createReadStream`, Atomic Rename & `stat` in real Node js systems — specifically: Usi...

4.4 `node:path` Security & Cross-Platform Paths — `join` vs `resolve`, `parse` & Blocking Path Traversal [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply `node:path` Security & Cross-Platform Paths — `join` vs `resolve`, `parse` & Blocking Path Traversal in real Node js systems — specifically: POSIX...

Module 4 Hands-On Capstone Project (100 Marks): Constant-Memory Streaming CSV/NDJSON Sanitizer & Path-Safe File Vault — Create a streaming Node.js ETL processor using `node:stream/promises` `pipeline()` and a custom `Transform` stream that redacts PII emails line-by-line from arbitrarily large log files in fl...

MODULE 5 OF 16 • 14 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Native HTTP Web Servers (`node:http`), Request/Response Streams, URL Query & `node:zlib`

Build zero-framework HTTP/1.1 servers with `node:http`, parse streaming JSON request bodies with size limits, query URLs via WHATWG `URL`/`URLSearchParams`, and compress responses using Gzip/Brotli (`node:zlib`).

Objectives: ✓ Understand `IncomingMessage` (Readable stream) and `ServerResponse` (Writable stream) lifecycles • ✓ Enforce strict payload byte limits & socket timeouts to block Slowloris/OOM attacks • ✓ Parse query parameters cleanly with WHATWG `URLSearchParams` and stream Brotli/Gzip compression

5.1 Core Web Server Module (`node:http`) — `createServer`, Keep-Alive Agents & Socket Timeouts [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Core Web Server Module (`node:http`) — `createServer`, Keep-Alive Agents & Socket Timeouts in real Node js systems — specifically: How `http.creat...

5.2 Handling HTTP Requests & Responses — Streaming Body Chunks, Status Codes & Header Safety [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Handling HTTP Requests & Responses — Streaming Body Chunks, Status Codes & Header Safety in real Node js systems — specifically: Reading chunked r...

5.3 URL & Query String Parsing — Modern WHATWG `URL` & `URLSearchParams` vs Legacy `node:querystring` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply URL & Query String Parsing — Modern WHATWG `URL` & `URLSearchParams` vs Legacy `node:querystring` in real Node js systems — specifically: Why WHAT...

5.4 HTTP & Stream Compression (`node:zlib`) — Gzip, Deflate, Brotli & Zip-Bomb Protection [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply HTTP & Stream Compression (`node:zlib`) — Gzip, Deflate, Brotli & Zip-Bomb Protection in real Node js systems — specifically: Compressing JSON/HTM...

Module 5 Hands-On Capstone Project (100 Marks): Zero-Dependency `node:http` JSON Micro-Router with Brotli Compression & Body Size Guard — Build a lightweight native `node:http` JSON server supporting `/health`, `/courses?category=...&page=...` via `new URL(req.url, base)`, bounded JSON body parsing (rejecting > 1 MB with `413...

MODULE 6 OF 16 • 12 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Applied Cryptography (`node:crypto`), Constant-Time HMAC Verification & TLS/SSL Security

Hash passwords safely with salted `crypto.scrypt`, encrypt secrets with authenticated `AES-256-GCM`, defeat timing side-channel attacks using `crypto.timingSafeEqual()`, and configure TLS 1.3 / mTLS servers.

Objectives: ✓ Use `crypto.scrypt` + `randomBytes` salts for passwords instead of fast SHA-256/MD5 hashes • ✓ Verify Razorpay/Stripe/GitHub HMAC-SHA256 webhook signatures in constant time with `timingSafeEqual` • ✓ Configure modern TLS 1.3 cipher suites, HSTS, and mutual TLS (`node:tls` / `node:https`)

6.1 `node:crypto` Essentials — `randomUUID`, Salted `scrypt` Passwords, `createHmac` & `AES-256-GCM` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply `node:crypto` Essentials — `randomUUID`, Salted `scrypt` Passwords, `createHmac` & `AES-256-GCM` in real Node js systems — specifically: CSPRNG to...

6.2 Constant-Time Secret Comparison (`crypto.timingSafeEqual`) — Defeating Timing Side-Channel Attacks [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Constant-Time Secret Comparison (`crypto.timingSafeEqual`) — Defeating Timing Side-Channel Attacks in real Node js systems — specifically: Why sta...

6.3 TLS/SSL & HTTPS in Node.js (`node:tls`, `node:https`) — TLS 1.3, Certificates, ALPN & mTLS [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply TLS/SSL & HTTPS in Node.js (`node:tls`, `node:https`) — TLS 1.3, Certificates, ALPN & mTLS in real Node js systems — specifically: TLS handshake f...

Module 6 Hands-On Capstone Project (100 Marks): Constant-Time Webhook Signature Verifier, `scrypt` Auth Vault & AES-256-GCM Token Engine — Implement a zero-dependency Node.js security library using `node:crypto` providing salted async `scrypt` password hashing, authenticated `aes-256-gcm` encryption (with 12-byte IV + 16-byte A...

MODULE 7 OF 16 • 12 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Multi-Core Scaling — `node:child_process`, `spawn` vs `fork` IPC & Custom NPM Modules

Scale beyond a single CPU core safely with `node:child_process` (`spawn`, `execFile`, `fork` IPC), avoid shell-injection vulnerabilities, and package clean dual/ESM custom Node.js modules.

Objectives: ✓ Prevent command injection by using argument-array `spawn`/`execFile` instead of shell-interpolated `exec` • ✓ Choose accurately between streaming `spawn()`, IPC-enabled `fork()`, and shared-memory `worker_threads` • ✓ Author reusable custom Node.js packages with clean `package.json` `"exports"` maps and encapsulation

7.1 `node:child_process` Security & Mechanics — `execFile`, `spawn`, Exit Signals & Avoiding Shell Injection [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply `node:child_process` Security & Mechanics — `execFile`, `spawn`, Exit Signals & Avoiding Shell Injection in real Node js systems — specifically: T...

7.2 `spawn()` vs `fork()` vs `worker_threads` — Stdio Streams, V8 Process Isolation & IPC Channels [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply `spawn()` vs `fork()` vs `worker_threads` — Stdio Streams, V8 Process Isolation & IPC Channels in real Node js systems — specifically: Comparing `...

7.3 Creating Custom Node.js Modules — `exports` Subpath Encapsulation, Barrel Pitfalls & Dependency Injection [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Creating Custom Node.js Modules — `exports` Subpath Encapsulation, Barrel Pitfalls & Dependency Injection in real Node js systems — specifically:...

Module 7 Hands-On Capstone Project (100 Marks): Multi-Process PDF/Report IPC Worker Pool (`fork`) & Reusable `@maa/logger` Custom Package — Build a bounded `child_process.fork()` worker pool with parent<->child IPC messaging, crash auto-respawn, and per-job timeouts so heavy report generation never blocks HTTP latency, packaged...

MODULE 8 OF 16 • 16 LIVE + LAB HOURS • 4 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Express.js Production REST API Engineering — Routing, Middleware, Request Validation & Uploads

Architect layered Express.js APIs with modular `express.Router()`, authentication/rate-limit/error middleware chains, schema-validated `req.params`/`req.query`/`req.body`, and hardened `multer` file uploads.

Objectives: ✓ Organize versioned `/api/v1` endpoints with modular `express.Router()` and proper HTTP status semantics • ✓ Master middleware execution order (`next()`, `next(err)`, 4-arg central error handler, Express 5 async rejection forwarding) • ✓ Validate untrusted `req.body`/`req.query` schemas and secure file uploads against MIME spoofing & disk exhaustion

8.1 Express.js Modular Routing — `express.Router()`, Route Params, Versioning & RESTful Status Codes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Express.js Modular Routing — `express.Router()`, Route Params, Versioning & RESTful Status Codes in real Node js systems — specifically: Mounting...

8.2 Express.js Middleware Pipeline — `(req, res, next)`, Execution Order, CORS, Helmet & 4-Arg Error Handler [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Express.js Middleware Pipeline — `(req, res, next)`, Execution Order, CORS, Helmet & 4-Arg Error Handler in real Node js systems — specifically: O...

8.3 Inspecting & Validating `req` in Express — `params`, `query`, `body`, `get()`, `app.set("trust proxy")` & Zod [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Inspecting & Validating `req` in Express — `params`, `query`, `body`, `get()`, `app.set("trust proxy")` & Zod in real Node js systems — specifical...

8.4 Secure File Uploads in Express (`multer` & Streaming Multipart) — Size Caps, UUID Names & Magic Bytes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Secure File Uploads in Express (`multer` & Streaming Multipart) — Size Caps, UUID Names & Magic Bytes in real Node js systems — specifically: Hand...

Module 8 Hands-On Capstone Project (100 Marks): Hardened Express.js Course & Assignment Portal REST API (`/api/v1`) — Build a production-structured Express.js REST API featuring modular `/api/v1/courses` & `/api/v1/submissions` routers, request-ID logging, Bearer JWT auth middleware, strict payload validati...

MODULE 9 OF 16 • 12 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Relational & Document Persistence — `mysql2/promise` Connection Pools & MongoDB Queries

Connect Node.js backends to MySQL/MariaDB with pooled prepared statements (`mysql2/promise`) and query MongoDB collections efficiently using compound indexes, projections, and cursor pagination.

Objectives: ✓ Eliminate connection-per-request churn and SQL Injection using `mysql2/promise` `createPool` + `?` prepared placeholders • ✓ Stream million-row SQL exports and release transactional `pool.getConnection()` handles inside `finally` blocks • ✓ Write selective, indexed MongoDB queries preventing NoSQL operator injection (`$ne`/`$gt`) and `COLLSCAN` table scans

9.1 MySQL/MariaDB in Node.js (`mysql2/promise`) — `createPool` vs `createConnection` & Transaction Safety [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply MySQL/MariaDB in Node.js (`mysql2/promise`) — `createPool` vs `createConnection` & Transaction Safety in real Node js systems — specifically: Why...

9.2 Querying MySQL Records Safely — `?` Prepared Placeholders, Keyset Pagination & Streaming Large Result Sets [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Querying MySQL Records Safely — `?` Prepared Placeholders, Keyset Pagination & Streaming Large Result Sets in real Node js systems — specifically:...

9.3 MongoDB Queries in Node.js — Compound Indexes (ESR Rule), Projections, Aggregations & Blocking NoSQL Injection [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply MongoDB Queries in Node.js — Compound Indexes (ESR Rule), Projections, Aggregations & Blocking NoSQL Injection in real Node js systems — specifica...

Module 9 Hands-On Capstone Project (100 Marks): Polyglot Analytics & Enrollment Data Layer (`mysql2` Transactions + MongoDB Indexed Queries) — Implement a transactional MySQL/MariaDB Enrollment Repository (`BEGIN` -> seat lock -> student group insert -> `COMMIT`/`ROLLBACK` with guaranteed `conn.release()` in `finally`) paired with...

MODULE 10 OF 16 • 12 LIVE + LAB HOURS • 2 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Native Automated Testing (`node:test`) & Production CI/CD Deployment Pipelines

Write fast zero-dependency unit & integration tests using Node's built-in `node:test` + `node:assert/strict` runner, mock clocks/timers cleanly, and ship zero-downtime Docker & GitHub Actions CI/CD pipelines.

Objectives: ✓ Write deterministic unit and HTTP integration tests using built-in `node:test` (`describe`, `it`, `mock`) and `node:assert/strict` • ✓ Test async rejections (`assert.rejects`) and time-dependent logic without flaky real-clock sleeps • ✓ Automate lint, `npm ci`, `npm audit`, matrix `node --test`, slim multi-stage non-root Docker builds, and health-checked rollouts

10.1 Automated Unit & Integration Testing (`node:test` & `node:assert/strict`) — Mocks, Async Rejections & Coverage [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply Automated Unit & Integration Testing (`node:test` & `node:assert/strict`) — Mocks, Async Rejections & Coverage in real Node js systems — specifica...

10.2 CI/CD & Production Node.js Operations — GitHub Actions, `npm ci --omit=dev`, Non-Root Docker & Health Probes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Purpose: To understand and apply CI/CD & Production Node.js Operations — GitHub Actions, `npm ci --omit=dev`, Non-Root Docker & Health Probes in real Node js systems — specificall...

Module 10 Hands-On Capstone Project (100 Marks): Zero-Flake `node:test` Suite, Code Coverage Gate & Multi-Stage Production CI/CD Workflow — Create a comprehensive `node --test --experimental-test-coverage` suite covering async services, mocked timers, and ephemeral-port HTTP endpoints using `node:assert/strict`, paired with a ha...

MODULE 11 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Production Auth, RBAC Permissions, Rate Limiting & OWASP Hardening

Production-grade Node js Training engineering module covering Production Auth, RBAC Permissions, Rate Limiting & OWASP Hardening with hands-on labs and capstone build.

Objectives: ✓ Short-Lived JWT Access + Rotating HttpOnly Refresh Tokens — Purpose: Prevent token theft and revoke compromised user sessions instantly across devices. • ✓ Role & Scope RBAC Middleware plus Sliding-Window Rate Limiting — Purpose: Enforce tenant isolation and block credential-stuffing or scraping floods at the edge. • ✓ Helmet Headers, Strict Schema Validation, CORS & HMAC Signatures — Purpose: Eliminate NoSQL/SQL injection, prototype pollution, and forged payment webhooks.

11.1 Short-Lived JWT Access + Rotating HttpOnly Refresh Tokens [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Short-Lived JWT Access + Rotating HttpOnly Refresh Tokens in real Node js systems — specifically: Prevent token theft and revoke compromised user...

11.2 Role & Scope RBAC Middleware plus Sliding-Window Rate Limiting [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Role & Scope RBAC Middleware plus Sliding-Window Rate Limiting in real Node js systems — specifically: Enforce tenant isolation and block credenti...

11.3 Helmet Headers, Strict Schema Validation, CORS & HMAC Signatures [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Helmet Headers, Strict Schema Validation, CORS & HMAC Signatures in real Node js systems — specifically: Eliminate NoSQL/SQL injection, prototype...

Module 11 Hands-On Capstone Project (100 Marks): Multi-Tenant Auth & HMAC Webhook Verification Gateway — Design, implement, test, and document 'Multi-Tenant Auth & HMAC Webhook Verification Gateway' applying every production concept from Module 11 (Production Auth, RBAC Permissions, Rate Limiti...

MODULE 12 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Redis Caching, Distributed Locks & BullMQ Background Worker Queues

Production-grade Node js Training engineering module covering Redis Caching, Distributed Locks & BullMQ Background Worker Queues with hands-on labs and capstone build.

Objectives: ✓ Cache-Aside Pattern, TTL Jitter & Targeted Key Invalidation — Purpose: Drop p95 database read latency from 120ms to 3ms on high-traffic endpoints. • ✓ BullMQ Job Producers, Concurrency Workers & Dead-Letter Queues — Purpose: Offload slow PDF/email/webhook work from HTTP threads with exponential retry safety. • ✓ Atomic Redis Locks & Idempotency-Key Payment Protection — Purpose: Guarantee retried POST requests never double-charge or double-book inventory.

12.1 Cache-Aside Pattern, TTL Jitter & Targeted Key Invalidation [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Cache-Aside Pattern, TTL Jitter & Targeted Key Invalidation in real Node js systems — specifically: Drop p95 database read latency from 120ms to 3...

12.2 BullMQ Job Producers, Concurrency Workers & Dead-Letter Queues [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply BullMQ Job Producers, Concurrency Workers & Dead-Letter Queues in real Node js systems — specifically: Offload slow PDF/email/webhook work from HT...

12.3 Atomic Redis Locks & Idempotency-Key Payment Protection [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Atomic Redis Locks & Idempotency-Key Payment Protection in real Node js systems — specifically: Guarantee retried POST requests never double-charg...

Module 12 Hands-On Capstone Project (100 Marks): Async Invoice, Email & Webhook Retry Engine with BullMQ — Design, implement, test, and document 'Async Invoice, Email & Webhook Retry Engine with BullMQ' applying every production concept from Module 12 (Redis Caching, Distributed Locks & BullMQ Ba...

MODULE 13 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Streams, Worker Threads, Cluster Mode & Graceful Shutdown

Production-grade Node js Training engineering module covering Streams, Worker Threads, Cluster Mode & Graceful Shutdown with hands-on labs and capstone build.

Objectives: ✓ stream.pipeline() Backpressure for Multi-GB Uploads & Exports — Purpose: Stream millions of records to S3 or HTTP responses without V8 heap exhaustion. • ✓ Worker Threads for CPU-Bound Crypto/Parsing & Cluster Scaling — Purpose: Utilize all CPU cores while keeping the main Node.js event loop < 10ms lag. • ✓ SIGTERM Graceful Drain, /healthz & /readyz Readiness Probes — Purpose: Deploy new Node releases with zero dropped requests or half-written transactions.

13.1 stream.pipeline() Backpressure for Multi-GB Uploads & Exports [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply stream.pipeline() Backpressure for Multi-GB Uploads & Exports in real Node js systems — specifically: Stream millions of records to S3 or HTTP res...

13.2 Worker Threads for CPU-Bound Crypto/Parsing & Cluster Scaling [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Worker Threads for CPU-Bound Crypto/Parsing & Cluster Scaling in real Node js systems — specifically: Utilize all CPU cores while keeping the main...

13.3 SIGTERM Graceful Drain, /healthz & /readyz Readiness Probes [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply SIGTERM Graceful Drain, /healthz & /readyz Readiness Probes in real Node js systems — specifically: Deploy new Node releases with zero dropped req...

Module 13 Hands-On Capstone Project (100 Marks): Constant-Memory Streaming CSV Importer & Multi-Core API — Design, implement, test, and document 'Constant-Memory Streaming CSV Importer & Multi-Core API' applying every production concept from Module 13 (Streams, Worker Threads, Cluster Mode & Grac...

MODULE 14 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Automated API Testing (Supertest), Structured Pino Logs & Docker CI

Production-grade Node js Training engineering module covering Automated API Testing (Supertest), Structured Pino Logs & Docker CI with hands-on labs and capstone build.

Objectives: ✓ Integration Testing Express Routers & Auth Flows with Supertest — Purpose: Verify status codes, DB side effects, and validation errors automatically on every PR. • ✓ AsyncLocalStorage Correlation IDs & Structured Pino JSON Logs — Purpose: Trace every log entry across async calls and redact sensitive secrets automatically. • ✓ Multi-Stage Node Alpine Dockerfiles & GitHub Actions Deployment — Purpose: Ship tiny, non-root Node.js production containers behind TLS reverse proxies.

14.1 Integration Testing Express Routers & Auth Flows with Supertest [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Integration Testing Express Routers & Auth Flows with Supertest in real Node js systems — specifically: Verify status codes, DB side effects, and...

14.2 AsyncLocalStorage Correlation IDs & Structured Pino JSON Logs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply AsyncLocalStorage Correlation IDs & Structured Pino JSON Logs in real Node js systems — specifically: Trace every log entry across async calls and...

14.3 Multi-Stage Node Alpine Dockerfiles & GitHub Actions Deployment [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Multi-Stage Node Alpine Dockerfiles & GitHub Actions Deployment in real Node js systems — specifically: Ship tiny, non-root Node.js production con...

Module 14 Hands-On Capstone Project (100 Marks): Containerized Node.js Microservice with Supertest CI Suite — Design, implement, test, and document 'Containerized Node.js Microservice with Supertest CI Suite' applying every production concept from Module 14 (Automated API Testing (Supertest), Struct...

MODULE 15 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Resilient Node.js Micro-APIs: Idempotency, Queues & Outbound Timeouts

Self-explanatory list-and-bullet production mastery module covering Resilient Node.js Micro-APIs: Idempotency, Queues & Outbound Timeouts in Node js Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Strict Request Schema Validation, Correlation IDs & Structured Pino JSON Logs — Purpose: Reject malformed payloads at the edge and trace every request across logs in milliseconds. • ✓ Outbound HTTP Timeouts, Circuit Breakers & Exponential Backoff Retries — Purpose: Prevent one slow third-party payment or SMS gateway from hanging your entire Node.js event loop. • ✓ Background Queue Workers, Dead-Letter Retries & Graceful SIGTERM Drain — Purpose: Finish in-flight jobs safely during container restarts without losing customer tasks. • ✓ Build, verify, and document 'Fault-Tolerant Webhook & Background Job Worker Service in Node.js' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

15.1 Strict Request Schema Validation, Correlation IDs & Structured Pino JSON Logs [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Strict Request Schema Validation, Correlation IDs & Structured Pino JSON Logs in real Node js systems — specifically: Reject malformed payloads at...

15.2 Outbound HTTP Timeouts, Circuit Breakers & Exponential Backoff Retries [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Outbound HTTP Timeouts, Circuit Breakers & Exponential Backoff Retries in real Node js systems — specifically: Prevent one slow third-party paymen...

15.3 Background Queue Workers, Dead-Letter Retries & Graceful SIGTERM Drain [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Background Queue Workers, Dead-Letter Retries & Graceful SIGTERM Drain in real Node js systems — specifically: Finish in-flight jobs safely during...

Module 15 Hands-On Capstone Project (100 Marks): Fault-Tolerant Webhook & Background Job Worker Service in Node.js — Architect, implement, test, and document 'Fault-Tolerant Webhook & Background Job Worker Service in Node.js' applying every bullet-wise engineering concept from Module 15 (Resilient Node.js...

MODULE 16 OF 16 • 14 LIVE + LAB HOURS • 3 TOPICS • 1 CAPSTONE PROJECT (100 MARKS)

Node.js Security Hardening, Integration Testing & Backend System Design

Self-explanatory list-and-bullet production mastery module covering Node.js Security Hardening, Integration Testing & Backend System Design in Node js Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken Gemini AI Viva.

Objectives: ✓ Preventing Prototype Pollution, NoSQL/SQL Injection, SSRF & Token Replay — Purpose: Lock down public Node.js APIs against the most common real-world attacker exploits. • ✓ Automated API Integration Testing with Node Test Runner / Vitest & Supertest — Purpose: Verify status codes, auth scopes, and database side effects automatically before deployment. • ✓ Event-Loop Latency Diagnostics, Memory Leaks & Node.js Interview Scenarios — Purpose: Spot blocked event loops, unbounded caches, and concurrency bottlenecks in live interviews. • ✓ Build, verify, and document 'OWASP-Hardened Express API Suite with Supertest Automated Verification' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + Google Gemini AI Technical Interviews.

16.1 Preventing Prototype Pollution, NoSQL/SQL Injection, SSRF & Token Replay [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Preventing Prototype Pollution, NoSQL/SQL Injection, SSRF & Token Replay in real Node js systems — specifically: Lock down public Node.js APIs aga...

16.2 Automated API Integration Testing with Node Test Runner / Vitest & Supertest [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Automated API Integration Testing with Node Test Runner / Vitest & Supertest in real Node js systems — specifically: Verify status codes, auth sco...

16.3 Event-Loop Latency Diagnostics, Memory Leaks & Node.js Interview Scenarios [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Purpose: To understand and apply Event-Loop Latency Diagnostics, Memory Leaks & Node.js Interview Scenarios in real Node js systems — specifically: Spot blocked event loops, unbou...

Module 16 Hands-On Capstone Project (100 Marks): OWASP-Hardened Express API Suite with Supertest Automated Verification — Architect, implement, test, and document 'OWASP-Hardened Express API Suite with Supertest Automated Verification' applying every bullet-wise engineering concept from Module 16 (Node.js Secur...

3. LIVE CLIENT PRODUCTION PROJECT, GLOBAL CERTIFICATIONS & CAREER SCOPE

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

Project: SwiftDispatch High-Throughput Logistics API Gateway & Webhook Processor • **Confirmed Role:** Node.js Backend API Engineering Intern

Domain & Production Infra: Supply Chain & Hyperlocal Delivery Client (Node.js + Express + Cluster Workers + PM2/Docker HTTPS Production Server)

- Engineered non-blocking Node.js & Express REST APIs with modular controllers, request validation, and centralized error middleware.
- Implemented secure JWT authentication, RBAC permissions, HMAC webhook signature verification, and rate limiting.
- Built asynchronous background job queues, streaming file processors, and connection-pooled database repositories.
- Deployed hardened Node.js services to live client production with structured JSON logging and uptime health checks.

Aligned Global Vendor Certifications: **JSNAD** (OpenJS Node.js Application Developer (JSNAD)) • **JSNSD** (OpenJS Node.js Services Developer (JSNSD)) • **C100DEV** (MongoDB Certified Associate Developer)

Target Job Roles: Node.js Backend Developer, REST API & Microservices Engineer, Backend SDE-1 (JavaScript/Node), Server-Side Application Engineer

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹5.0 LPA – ₹9.5 LPA **Mid-Level Node.js Backend Engineer (1–3 Yrs):** ₹10.0 LPA – ₹20.0 LPA **Senior Backend / Systems Architect (3–6+ Yrs):** ₹20.0 LPA – ₹38.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/node-js-training

Try 2 Free Topic-Wise 50-MCQ Mock Tests & Spoken Gemini AI Interviews daily on every topic above before enrolling.