

Backend Engineering

Duration	Live classes	Schedule	Batch size
10 weeks	100 hours	Mon to Fri, 2 hrs/day	10 students max
Tools you will use			
Node.js + TypeScript NestJS + Prisma PostgreSQL + Redis System Design Docker + DevOps			

10 students a batch. Exceptional mentors who guide you personally, and a **syllabus built from real job posts**, cuz we never skip the "hard parts" 😊

The class

10 per batch

Ten students. Never more.

A hard cap, not a target. Your mentor knows your name and your work.

The mentor

4–5 yrs in the job

Someone who does the job.

Working professionals, not full-time teachers. They read your work line by line.

The syllabus

1/3 is fundamentals

The hard parts, not just the basics.

Teaches everything the job needs and never skips the hard parts most courses skip, built from real job descriptions. Check the syllabus before you enroll.

Built into every course

Gate assessment

A short test before Day 1 finds your strengths and pace, then sets your plan.

A real, live project

Every course ends with a deployed project you can show employers.

Revision every 5th class

Every fifth class goes back over the week, so nothing piles up.

1:1 mock interview

A real interview practice with your mentor in the final week.

Certification exam

Pass the exam at the end of the course to earn your certificate.

CV and LinkedIn help

We help you rewrite your CV and LinkedIn before you apply.

Backend Engineering

Why This Course

This course takes you from your first line of code to mid-level backend skills, plus senior-level system design. The backend is the part of an app you do not see: the servers, APIs and databases behind it.

- **You learn what employers want.** They want people who build clean APIs (the way apps talk to a server) and store data well in PostgreSQL (a database). They also want people who can grow an app for more users and run it on live servers.
- **You go deep in one set of tools.** You learn JavaScript and TypeScript, then Express and NestJS. Then you spend two weeks on databases with PostgreSQL, Prisma and Redis.
- **You also learn senior topics.** These include security and queues (jobs that run in the background). You get a full week of system design (planning large apps) and a full week of DevOps (putting apps online and keeping them running).
- **Be honest about the market.** Many Kathmandu companies use PHP and Laravel. Node.js jobs pay best on remote teams abroad.
- **You finish with a live project.** Most job ads ask for 1 year of experience. A working project with a design document and its code on GitHub helps fill that gap.
- **You code with AI tools like Cursor and Copilot.** You must check and understand every line the AI writes.
- **We do not promise a salary.** Ten weeks teaches you the topics. Becoming a senior engineer still takes years of work.

Who Is This Course For

- **Complete beginners** who want a real backend career and will put in the work on the basics.
- **Frontend or JavaScript learners** who want to learn the server side and become full stack developers.
- **Junior developers** who can build simple apps but cannot yet explain indexes, transactions, scaling or deployment.
- **Self-taught coders** who have never planned an API, a database design, a login system or a large multi-server system.
- **Anyone** who wants to go deep in Node.js, PostgreSQL, system design and DevOps, and put real apps online.

Course Snapshot

PARAMETER	DETAILS
Course Code	TR-12
Duration	10 Weeks (2.5 Months)
Schedule	Monday to Friday, 2 Hours/Day
Total Hours	100 Hours
Batch Size	Maximum 10 Students
Course Fee	NPR 36,750
Level	Beginner start, taught from the fundamentals → mid/senior-level depth
Outcome	Backend Engineer (Node.js, TypeScript, NestJS, PostgreSQL, system design, Docker and DevOps, deployed to production), ready for Nepal and remote teams

Prerequisites

- You need no coding experience. We teach JavaScript and TypeScript from zero in Weeks 2 and 3.
- You need a laptop with at least 8GB RAM. Any recent Windows, macOS or Linux laptop works.
- You use a computer every day and know how files and folders work.
- You can read and write basic English. Clear written English is what gets you hired by remote teams.
- You take the Saarathi Gate Assessment before Day 1. It only checks your starting level, so there is no pass or fail.

You do not need to install anything first. We set up VS Code, Node.js, Git, GitHub and Docker together in the first weeks.

Assessments at a glance

- **Gate Assessment before Day 1.** It has no pass or fail.
- **Certification Exam.** It lasts 2 hours and has about 30 questions of different types. You need 70% to pass and get your certificate.

- **Exam rules.** The exam runs in fullscreen. Switching tabs and pasting are recorded, and three of these end your exam. Answer every question.
- **Recommendation letters are not automatic.** Mentors give them only to learners who pass, and they are never guaranteed.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class with your mentor, where you build real projects
Mon–Fri	2 hours of study at home to build features, fix errors and read documentation
Sunday	Open classroom if you want to build, or rest
Day 5 (every week)	The last hour of class is for revision, a review of how your project is built, and clearing doubts

Your mentor gives you 2 projects during the course. They need to be approved before you get your certificate and before the Career Launchpad opens.

You improve most by building at home between classes, making mistakes and fixing them.

How the Difficulty Is Paced

You learn at most **seven topics a week**. You never get two new hard topics at the same time.

- **Weeks 1 to 4 are the slow start.** You learn one idea at a time. You spend two full weeks on JavaScript and TypeScript, then build your first API with Express.
- **Weeks 5 to 7 get harder.** You learn NestJS, then spend two weeks on databases. Your app moves to PostgreSQL without a rewrite.
- **Week 8 makes your app ready for real users.** You add login, security, background jobs, live updates and AI features to an app you already know.
- **Weeks 9 and 10 cover senior skills.** You spend one week on system design, then one week on DevOps to put everything online.

Each week has a checkpoint and a list of things to build between classes. With only ten students in a batch, your mentor can follow every student's progress.

Week-by-Week Curriculum

Phase 1 · Web, JavaScript, TypeScript, Node.js & REST APIs (Weeks 1–4, 40 Hours)

WEEK	FOCUS AREA	TOPICS
Week 1	How the Web Works, the Linux Terminal & Git	- What backend engineering is: frontend vs backend vs database, the client-server model; the course model named up front (request → route → validate → logic → database → response) - One request end to end: DNS, TCP, TLS/HTTPS, the HTTP round trip, and HTTP/1.1 vs HTTP/2 vs HTTP/3 - HTTP in depth: methods, status codes, headers, cookies and caching headers, inspected with dev-tools and <code>curl</code> - REST: resources as URLs, verbs as actions, statelessness; GraphQL and gRPC named and compared - The Linux terminal: files, paths, pipes, permissions, processes, ports, environment variables, SSH keys - Git and GitHub from zero: commit, branch, pull request, merge conflicts and code review - Your machine set up once: VS Code, Node.js, Git, GitHub, plus the Cursor/Copilot verify-discipline rule set
Week 2	JavaScript Foundations	- Values, variables (<code>let</code> / <code>const</code>), types, template literals, operators and truthiness - Control flow: <code>if</code> / <code>else</code> and loops (<code>for</code> , <code>while</code> , <code>for...of</code>) - Functions: parameters, return values, scope, closures, arrow functions - Arrays and <code>map</code> / <code>filter</code> / <code>reduce</code> ; <code>Map</code> and <code>Set</code> , and Big-O at intuition level - Objects, destructuring and spread, classes and <code>this</code> , and JSON, the format every API speaks - Reading errors and debugging: stack traces, the classic beginner errors, the VS Code debugger - Problem-solving drills for the written test: strings, arrays, two pointers, frequency maps
Week 3	Async JavaScript, TypeScript & the Node.js Runtime	- Error handling: <code>try</code> / <code>catch</code> , <code>Error</code> objects, why silent failures kill backends - Callbacks, Promises and <code>async</code> / <code>await</code> ; <code>Promise.all</code> vs <code>allSettled</code> , unhandled rejections - Node.js as a runtime: the event loop, the thread pool, what blocks it, ES modules, npm and <code>.env</code>

WEEK	FOCUS AREA	TOPICS
		- TypeScript basics: types, interfaces, unions, narrowing, <code>unknown</code> vs <code>any</code> - TypeScript in depth: generics, utility types, <code>strict</code> mode, <code>tsconfig</code>, type-checking with <code>tsc</code> - Streams, buffers and file I/O; graceful shutdown - Your first HTTP server in TypeScript with Node's built-in <code>http</code> module (before the framework)
Week 4	Express & REST API Design	- Express (v5) in TypeScript: the app object, <code>Router</code>, route params, query strings - The middleware pipeline and layered architecture: routes, controllers, services and repositories (in-memory first) - Validation with Zod, central error handling and one consistent error format - API design: naming, status codes, versioning, filtering and sorting, cursor vs offset pagination - Idempotency keys and safe retries, ETags and conditional requests - Twelve-factor config, structured logging with pino, request IDs, a Postman collection - Building routes with AI, and the review checklist: input validated, auth and ownership checked, no hidden N+1, errors handled, no leaked secret

Phase 2 · NestJS & Databases (Weeks 5–7, 30 Hours)

WEEK	FOCUS AREA	TOPICS
Week 5	NestJS Architecture	- Why NestJS (v11): modules, controllers and providers, the Nest CLI and project layout - Dependency injection and inversion of control, provider scopes, the "can't resolve dependencies" error fixed on purpose - The request lifecycle: DTOs and pipes, guards, interceptors and exception filters - Config module and OpenAPI/Swagger docs generated from your DTOs - SOLID and the design patterns that matter in a backend: repository, strategy, factory, decorator - Modular monolith vs microservices, bounded contexts, Nest microservice transports

WEEK	FOCUS AREA	TOPICS
		- The week's build: your Express API rebuilt in NestJS on in-memory repositories
Week 6	SQL, PostgreSQL, MongoDB, Data Modeling & Prisma	- Why a database: relational vs NoSQL, chosen by access pattern - PostgreSQL run in Docker, the <code>psql</code> client and a GUI client - Core SQL: CRUD, JOINS, aggregates, <code>GROUP BY</code>, subqueries, CTEs and window functions; SQL injection and parameterised queries - Data modeling: entity-relationship diagrams, normalisation to 3NF, when to denormalise, constraints - Prisma: schema, <code>migrate dev</code> vs <code>migrate deploy</code>, the typed client, relations, nested writes, raw SQL with <code>\$queryRaw</code>; TypeORM awareness - Swapping your in-memory repositories for Prisma in the NestJS app - MongoDB: documents and collections, CRUD, embedding vs referencing, the aggregation pipeline, indexes, Mongoose in NestJS
Week 7	Database Internals, Performance & Caching	- Indexes: B-tree, composite column order, covering and partial indexes, GIN for JSONB and full-text search - Reading query plans with <code>EXPLAIN ANALYZE</code>, the N+1 trap, Prisma query logging, connection pooling with PgBouncer - Transactions and ACID, isolation levels and MVCC - Locking and race conditions: lost updates, <code>SELECT ... FOR UPDATE</code>, optimistic locking, deadlocks - Unique-constraint conflicts returned as a clean <code>409</code>, and zero-downtime migrations (expand and contract) - Redis: data structures, cache-aside, TTL, invalidation and cache stampedes - Redis beyond caching: rate limiting, sessions and distributed locks

Phase 3 · Security, Async & AI Backends (Week 8, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 8	Auth, Security, Queues, Real-Time & AI in the	- Authentication : password hashing with bcrypt/argon2 , sessions vs JWT , refresh-token rotation and revocation

WEEK	FOCUS AREA	TOPICS
	Backend	- Authorisation: RBAC, object-level checks as NestJS guards, OAuth2/OpenID Connect "login with Google" - OWASP API Security Top 10 (BOLA first), Helmet, CORS, rate limiting, secrets management, file uploads with presigned URLs to S3-compatible storage - Background jobs with BullMQ: retries with backoff, dead-letter queues, idempotent consumers, the transactional outbox - Event-driven systems: Kafka and RabbitMQ concepts, topics, partitions, consumer groups, delivery guarantees - Real-time: WebSockets with NestJS gateways, Server-Sent Events, scaling sockets with Redis - AI in the backend: an LLM API call (streaming, timeout, cost, validated output) and a RAG endpoint with embeddings in pgvector

Phase 4 · System Design (Week 9, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 9	System Design: From One Server to Millions of Users	- Fundamentals: latency vs throughput, back-of-the-envelope estimation, availability, SLA/SLO/SLI - Scaling: vertical vs horizontal, stateless services, load balancers (L4/L7), API gateways, consistent hashing - Caching and CDNs at scale; database scaling: replication and lag, partitioning and sharding - Consistency: CAP and PACELC, strong vs eventual, consensus (Raft) at concept level, two-phase commit vs sagas - Reliability: timeouts, retries with jitter, circuit breakers, bulkheads, backpressure, rate-limiter algorithms - Architecture: monolith vs microservices, service boundaries, CQRS and event sourcing, service discovery - Design real systems in the interview format (URL shortener, chat, news feed, payments) and write a design doc

Phase 5 · DevOps, Deployment & Career (Week 10, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 10	DevOps, Production & Getting Hired	- Docker: images, layers, multi-stage builds, non-root containers; Docker Compose with app, PostgreSQL and Redis - A Linux VPS: SSH hardening, firewall, Nginx reverse proxy, Let's Encrypt HTTPS, a public <code>https://</code> URL - CI/CD with GitHub Actions: lint, type-check, build, image push, deploy on <code>main</code>, rollback; blue-green and canary releases - Cloud and orchestration: AWS core (EC2, S3, RDS, IAM), Kubernetes concepts (pods, deployments, services), Terraform infrastructure as code - Observability: structured logs, metrics with Prometheus and Grafana, traces with OpenTelemetry, alerts on SLOs; <code>pg_dump</code> backups with a practised restore - Live-debugging drills, timed and AI allowed: a broken endpoint, a slow query and a failing deploy; then capstone sign-off - Resume, GitHub and LinkedIn, a mock interview loop (API design, SQL, system design, debugging), target-company list and outreach

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Web & Protocols	HTTP/1.1, HTTP/2, HTTP/3, DNS, TCP, TLS/HTTPS, REST, GraphQL and gRPC awareness
Language & Runtime	JavaScript, TypeScript (generics, strict mode), Node.js (event loop, streams, async/await), Linux terminal, Git
API Design	Express (v5), Zod validation, error contracts, versioning, cursor pagination, idempotency keys, OpenAPI/Swagger
NestJS & Architecture	NestJS (v11): modules, dependency injection, guards, pipes, interceptors; SOLID, design patterns, modular monolith vs microservices
Databases	PostgreSQL, SQL (JOINS, CTEs, window functions), data modeling, Prisma, indexes, EXPLAIN ANALYZE, transactions, isolation levels, locking, MongoDB, Mongoose

CATEGORY	TECHNOLOGIES
Caching & Async	Redis (caching, rate limiting, locks), BullMQ queues, transactional outbox, Kafka and RabbitMQ concepts, WebSockets
Auth & Security	bcrypt/argon2, sessions, JWT, OAuth2/OpenID Connect, RBAC, OWASP API Security Top 10, Helmet, rate limiting
AI-Backend Integration	LLM API calls (streaming, timeouts, cost, output validation), RAG with embeddings and pgvector
System Design	Estimation, load balancing, caching/CDNs, replication, sharding, CAP/PACELC, sagas, circuit breakers, CQRS, design docs
DevOps	Docker, Docker Compose, Nginx, Let's Encrypt, GitHub Actions CI/CD, AWS core, Kubernetes concepts, Terraform, Prometheus, Grafana, OpenTelemetry
AI Productivity	Cursor, GitHub Copilot (with verify-discipline)

Topic Depth and Awareness

Depth: practised until you can apply it. **Awareness:** introduced so you can read and compare it later.

FOCUS AREA	LEARNING FOCUS
Web fundamentals (HTTP, DNS, TLS, REST)	Depth
Linux terminal and Git (branching, pull requests, code review)	Depth
JavaScript (values, functions, closures, arrays, objects, JSON)	Depth, one full week
Async JavaScript, TypeScript and the Node.js runtime	Depth, one full week
Reading errors and debugging	Depth, drilled all course
Express and REST API design (validation, errors, pagination, idempotency)	Depth
NestJS (modules, DI, DTOs, pipes, guards, interceptors) and design patterns	Depth, one full week

FOCUS AREA	LEARNING FOCUS
SQL, PostgreSQL, data modeling and Prisma	Depth, one full week
MongoDB (documents, embedding vs referencing, aggregation pipeline, Mongoose)	Hands-on
Database internals (indexes, EXPLAIN, transactions, isolation, locking) and Redis	Depth, one full week
Authentication, authorisation and OWASP API Security Top 10	Depth
Background jobs (BullMQ, outbox, idempotent consumers)	Hands-on
WebSockets and real-time	Hands-on
LLM API calls and a RAG endpoint (pgvector)	Hands-on
System design (scaling, caching, replication, sharding, consistency, reliability)	Depth (concept), one full week
Designing a real system end to end and writing a design doc	Depth, guided designs
Docker, Compose, VPS, Nginx/HTTPS, GitHub Actions CI/CD, observability	Depth
AWS core, Kubernetes, Terraform	Working level
Kafka / RabbitMQ, consensus (Raft), event sourcing	Concept level
TypeORM, GraphQL, gRPC	Awareness

Project Pool

In Week 5, you pick **one** final project (capstone) with your mentor. By Week 10 you build it, secure it, make it handle more users and put it online, with a design document. The list changes from batch to batch:

#	PROJECT	STACK FOCUS
1	Task/project management API with users, teams and role-based access	NestJS, Prisma, JWT, RBAC, PostgreSQL

#	PROJECT	STACK FOCUS
2	E-commerce backend: products, cart, orders, payments-stub, inventory	NestJS + Prisma, transactions, locking, Redis cache, outbox
3	Booking/reservation system with availability and double-booking prevention	NestJS, isolation levels, idempotency keys, WebSockets
4	Content/blog platform API with auth, media uploads and search	NestJS, presigned uploads, Postgres full-text search
5	Real-time chat or notifications service	NestJS gateways, Redis, BullMQ, horizontal scaling
6	URL shortener or analytics API at scale	NestJS, Redis, rate limiting, indexing, load testing

Career Paths & Trajectory

ROLE	TIMELINE	FOCUS
Junior Backend Developer / Backend Intern	First 3–6 months	Build and fix API features with a senior checking your work, and use your portfolio to land a first job
Backend Developer / Node.js Developer	0–2 yrs	Own features across API, database and auth in a real product
Backend Engineer / Full Stack Engineer	2–4 yrs	Design services, model data at scale, own performance and reliability
Senior Backend Engineer	4–7 yrs	System design, distributed systems, mentoring
Staff Engineer / Backend Architect / Platform Engineer	7+ yrs	Architecture, org-wide standards, scale and cost

FAQ

What does the Backend Engineering course cost and how do I pay?

NPR 36,750. Pay by cash at our Old Baneshwor campus, bank transfer, or Fonepay. Email info@saarathiacademy.com.np for the payment plan.

Do I need prior experience?

No coding experience is needed. You need to be comfortable using a computer, and you need a laptop. The Gate Assessment before Day 1 shows your starting level, and there is no pass or fail.

Is this a junior course or does it go further?

It starts from zero and goes up to topics asked in mid and senior interviews. Your job title will still depend on your real work experience.

I have never coded. Will I be able to keep up?

Yes. The course is paced for people who have never coded.

- You learn at most seven topics a week, and you get two full weeks on JavaScript and TypeScript.
- Each week has a checkpoint and a practice list.
- You get a revision hour every Day 5, a Sunday open classroom, and a batch of only ten students.

How long is the course?

10 weeks (2.5 months). You build your final project during the course and put it online. In the final week you do a mock job interview.

Which stacks and tools will I actually use?

- **Node.js, TypeScript, Express, NestJS**
- **PostgreSQL, Prisma, Redis, BullMQ**, WebSockets, JWT, OAuth2
- An **LLM API** and a small **RAG** endpoint with pgvector
- **Docker, GitHub Actions**, Nginx/HTTPS, **AWS, Kubernetes** concepts, **Terraform**, Prometheus, Grafana, OpenTelemetry

Why one stack instead of many?

If you only touch five frameworks quickly, you learn none of them well. Going deep in one set of tools leaves time for the language, the database, system design and DevOps.

What does a backend interview in Nepal actually look like?

It depends on the size of the company. We prepare you for all three:

- **Large companies** give an online coding test, then a technical interview, then a technical or design round.
- **Mid-sized companies** are most of the market. They give a 1 to 2 hour written test at their office, then an interview.
- **Startups** start with a chat with the founder or team lead, then give a task to finish at home in 1 to 3 days.

The written tests are basic. You solve string and array problems by hand and answer questions on SQL joins and normalisation (organising database tables). We practise these in Weeks 2 and 6. Week 10 has a full set of mock interviews.

What roles does this prepare me for?

See Career Paths above. We do not promise jobs.

- We help you build your portfolio, practise interviews, and improve your CV, GitHub and LinkedIn.
- We refer you to companies in our hiring network.
- Starting pay in Nepal is modest. Remote teams abroad pay more.
- Remote teams want projects they can run and clear written English. The course builds both.

How does the batch and mentorship work?

- Each batch has at most 10 students, so you get direct time with your mentor and feedback on your projects.
- Classes are in person at Old Baneshwor. Recordings and extra help hours are online.
- The Sunday Open Classroom is free for enrolled students.

Enrollment

- **Next Batch:** see the live course page
 - **Location:** Old Baneshwor Chowk, Kathmandu, Nepal
 - **Mode:** Online + Offline
 - **Contact:** 9744442469, 9761095364
 - Limited to 10 seats per batch
-

Backend Engineering — full details, fees and next batch dates:
saarathiacademy.com.np/backend-engineering-course-in-nepal

Version 4.0 · Updated 16 Ashwin 2083

