Saarathi Academy

TR-12 Course Brief

Version 1.2.0 · Mangsir 2083

Backend Engineering with AI

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 · Express + NestJS · PostgreSQL + Prisma + Redis · Auth + Queues + RAG · System Design · Docker + GitHub Actions

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.
- **Revision every 5th class**
  Every fifth class goes back over the week, so nothing piles up.
- **Certification exam**
  Pass the exam at the end of the course to earn your certificate.
- **A real, live project**
  Every course ends with a deployed project you can show employers.
- **1:1 mock interview**
  A real interview practice with your mentor in the final week.
- **CV and LinkedIn help**
  We help you rewrite your CV and LinkedIn before you apply.

saarathiacademy.com.np

Old Baneshwor Chowk, Kathmandu

+977-9744442469 +977-9761095364 +977-9768618434

Backend Engineering with AI

Why This Course

This course takes you from your first line of code to a junior backend developer who can build AI features into a backend, with the database and system design depth mid-level interviews ask. 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), store data well in PostgreSQL (the database most backend jobs ask for), and run it on a live server.
- **You go deep in one set of tools.** You learn JavaScript and TypeScript together in one week, then Node.js, then Express. Your API then gets a real database with PostgreSQL and Prisma, a team structure and real login in NestJS 12, then Redis, background jobs and Docker.
- **You learn to read code before you write it.** AI tools write much of the code on a real team now. So you learn to predict what code does, trace it, explain it in plain English and check what an AI wrote.
- **You put your work online early.** Your first deploy is in Week 4, on your own subdomain of the academy's batch server. Week 10 automates it with CI/CD.
- **You also learn production topics.** Email verification and forgot password, permissions and security, queues (jobs that run in the background), webhooks, scaling to many servers, and health checks and backups. Tools for much larger systems, such as Kubernetes and Terraform, are only named.
- **You build AI features into your backend.** You call an AI model safely, let it use your own API's functions (tool calling), search your data by meaning with pgvector, and answer questions from your own documents (RAG). Then you guard, test and cost what you built, and design a whole system with a payment and an AI feature in it.
- **Be honest about the market.** Many Kathmandu companies use PHP and Laravel. Node.js jobs pay best on remote teams abroad.
- **You finish with live work online.** Most job ads ask for 1 to 2 years of experience. A working backend 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 deploy.
- **Anyone** who wants to go deep in Node.js, PostgreSQL, AI features and system design, 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 → junior backend developer who builds AI features, with the database and system design depth mid-level interviews ask
Outcome	Junior Backend Developer (Node.js, TypeScript, NestJS, PostgreSQL, Docker, deployed live) who can add AI features (LLM calls, tool calling, RAG with pgvector), with the database and system design depth mid-level interviews ask, for Nepal and remote teams

Prerequisites

- You need no coding experience. We teach JavaScript and TypeScript from zero in Week 2, and Node.js in Week 3.
- You need your own laptop. 8GB of RAM works if you run one database at a time. 16GB is more comfortable for Weeks 7 to 10.

- 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 and GitHub together in Week 1, and you install Docker Desktop as Week 1 homework.

What it costs besides the fee. Nothing is required. Every tool in the course has a free version.

- **Server.** The academy runs one shared server per batch. Each student gets their own login and subdomain on it, so you never need a card to buy a server.
- **Domain.** Optional. Your work runs on your academy subdomain.
- **AI key.** The AI features in Weeks 8 and 9 use a free tier, such as the Gemini API or Groq. Never send private data on a free tier. A paid key is optional.
- **Email.** The emails your app sends in Week 8 use a free tier, or a mail catcher on your laptop.
- **Cloud accounts.** You do not need an AWS or Azure account. The mentor shows how your stack maps to AWS in Week 10.

Assessments at a glance

- **Gate Assessment.** You take it before Day 1. It is not pass or fail. It shows us how you learn and how fast.
- **Certification Exam.** You take it online at home on your own laptop in the final week. It lasts 2 hours and has around 30 questions of different types, including code and short real-world cases.
- **Pass mark.** You need 70 percent overall and at least 50 percent in every area. You must pass to get the Certificate of Completion.
- **Attendance.** You need at least 70% attendance to sit the exam.
- **Camera.** Your camera stays on for the full 2 hours.
- **Three strikes.** Three strikes end the exam early: leaving the exam tab, someone else in the camera, or a phone in view.
- **Recorded, not a strike.** Leaving fullscreen, pasting text and looking away are recorded for the Saarathi team, but they do not end the exam.
- **Checked by a person.** A person on the Saarathi team checks the camera pictures before a pass counts.
- **Answer order.** You can answer in any order, but every question needs an answer before you submit.
- **Mock interview.** Every student does one mock interview in Week 10. It has no pass or fail; it is completed with written feedback.

- **Recommendation letters.** We may give one to learners who pass, but it is never guaranteed.

Projects

Your mentor gives you 2 projects during the course, and they need to be approved before you get your certificate. The Career Launchpad stays locked until then.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class with your mentor: you predict what code does, run it, look into why, change it, then make a small piece yourself
Mon–Fri	2 hours of study at home for reading and explaining code, fixing errors, building endpoints and reading docs (more in Weeks 5, 8, 9 and 10)
Sunday	Open classroom if you want to build, or rest
Day 5 (every week)	The last hour of class is for revision. It opens with a 10-minute check with AI tools off, then a review of the week's code and clearing doubts

> You learn most by reading, running, breaking and fixing code between classes.

How the Difficulty Is Paced

You learn at most **seven topics a week**, and we aim for one hard new idea at a time. Every new idea arrives on the day you first need it.

- **Weeks 1 to 4 are the base.** Week 1 shows how a computer runs a program and how the web moves a request. Week 2 is the core of JavaScript and TypeScript, taught together. Week 3 is Node.js and asynchronous code in its own week, because beginners find it hardest. In Week 4 you build your first API with Express and put it online for the first time.
- **Weeks 5 to 7 go deeper.** Week 5 gives your working API a real PostgreSQL database and ends with indexes. In Week 6, NestJS gives the same API the structure larger teams use, and you add real login. Week 7 reads query plans, keeps data correct when many users buy at once, adds Redis and background jobs, and adds Docker, so the whole app runs with one command.
- **Week 8 makes your app ready for real users.** You add email verification and forgot password, permissions, security, webhooks, live updates and your first AI model call to an app you already know.

- **Week 9 is AI backends and system design, one idea a day.** Tool calling, then search by meaning, then RAG (answering from your own documents), then guarding and testing your AI feature, then scaling to many servers and designing one real system.
- **Week 10 puts it all live and gets you ready to apply.** Two days put your app live on the batch server with automatic deploys. The last three days are your CV and job applications, a mock interview and your project showcase.

You read code before you write it. Each class hour follows five steps: predict, run, look into it, change it, then make a small piece yourself. From Week 4, every build goes in as a pull request, and you review one classmate's pull request and one AI-written one each week.

Each week has a checkpoint and a list of things to practise 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 Computers & the Web Work, the Linux Terminal & Git	- What backend engineering is: frontend vs backend vs database, the client-server model; how a computer runs a program (**CPU**, memory and disk, a **process**, a **thread** and a **port**); the course model named up front (**request → route → validate → logic → database → response**) - One request end to end: **DNS**, **TCP**, **TLS/HTTPS** and the **HTTP** round trip; HTTP/2 and HTTP/3 at awareness - **HTTP** in depth: methods, status codes, headers, cookies and caching headers, inspected with dev-tools and `curl` - **REST**: resources as URLs, verbs as actions, statelessness; **GraphQL** and **gRPC** named - 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** (24 or 26 LTS), **Git**, **GitHub**, **Docker Desktop** as homework, plus the **Cursor**/**Copilot** rules: ask the AI why, and read every line before you accept it
Week 2	JavaScript & TypeScript, the Core	- Values, variables (`let` / `const`) and types in `.ts` files from the first line, with type inference; template literals, operators and truthiness - Control flow (`if` / `else` , `switch` , loops with `for` , `while`

WEEK	FOCUS AREA	TOPICS
		and `for...of`) and **tracing** with a trace table and the VS Code debugger - **Functions**, scope and the **call stack**: parameters, return values, arrow functions, reading a stack trace and the classic beginner errors, off-by-one among them - Arrays with `map` and `filter` (`reduce` read), objects, destructuring and spread, shared references, `Map` and `Set`, and **JSON**, the format every API speaks - **TypeScript** essentials: type annotations, `interface` vs `type`, optional fields, unions and narrowing - Working with AI on code: a clear task, asking why first, checking the answer by running and tracing it, and reviewing AI-written code line by line
Week 3	Node.js, Async Code & Your First Server	- ES modules, **npm**, `package.json`, the lockfile and `.env` - **Closures**, callbacks, **Promises** (`.then()` / `.catch()` read) and `async` / `await`: `try` / `catch`, `Error` objects, `Promise.all` vs `allSettled`, unhandled rejections, why silent failures kill backends - Classes, `this` and interfaces, with custom error classes - **Node.js** as a runtime: stack and queue as pictures, the **event loop** (timers, `setImmediate`, microtasks), the thread pool and what blocks it - Calling an outside API with `fetch`: the status code, checking the shape of the answer, a timeout, and reading generics such as `Promise<T>` - Your first HTTP server in TypeScript with Node's built-in `http` module (before the framework); file I/O with `fs/promises`, and streams named
Week 4	Express, REST API Design & Your First Deploy	- **Express** (v5) in TypeScript: the app object, `Router`, route params, query strings - The **middleware** pipeline and layered architecture: routes, controllers, services and **repositories** (in-memory first), a generic `Repository<T>` or `ApiResponse<T>`, and utility types such as `Record` and `ReturnType` - Validation with **Zod**, `unknown` vs `any`, discriminated unions, `strict` mode in `tsconfig` checked with `tsc`, central **error handling** and one consistent error format - **API design**: naming, status codes, **versioning**, filtering and sorting, **cursor vs offset pagination**; ETags at awareness - Twelve-factor config, structured logging with **pino**, request

WEEK	FOCUS AREA	TOPICS
		IDs, a **Bruno** collection kept in your repo - **Building routes with AI, and the review checklist** (input validated, auth and ownership checked, no hidden N+1, errors handled, no leaked secret), and every build sent in as a **pull request** - **Your first deploy**: the API online under your own login and subdomain on the academy's batch server, over HTTPS

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

WEEK	FOCUS AREA	TOPICS
Week 5	SQL, PostgreSQL, Data Modeling, Prisma & Indexes	- Why a database: **SQL and NoSQL**, and where a document database like **MongoDB** fits, chosen by how the data is read; MongoDB and MySQL at awareness - **PostgreSQL** run in **Docker**, the `psql` client and a GUI client - Core **SQL**: CRUD, **JOINs**, aggregates, `GROUP BY`, subqueries and **CTEs**; **SQL injection** and parameterised queries - **Data modeling**: entity-relationship diagrams, **normalisation** to 3NF, when to denormalise, **constraints** - **Prisma 7**: schema, `prisma.config.ts`, the Postgres driver adapter, `migrate dev` vs `migrate deploy`, running `prisma generate` yourself, the typed client and `$queryRaw`; **TypeORM** awareness - Swapping your in-memory repositories for **Prisma** in the Express app - **Indexes**: a tree and **Big-O** as pictures, then B-tree, composite column order and partial indexes, with a first `EXPLAIN ANALYZE` to prove an index is used; **window functions** at awareness
Week 6	NestJS Architecture & Login	- Why **NestJS** (v12): **modules**, **controllers** and **providers**, the Nest CLI, ES module packages - **Dependency injection** and inversion of control, a custom provider with an injection token (`useClass`), the "can't resolve dependencies" error fixed on purpose, **SOLID** and the repository, strategy, factory and decorator patterns; provider scopes at awareness - The request lifecycle: **pipes** with **Zod** schemas passed to `@Body()` and `@Query()` and checked by Nest's built-in validation pipe (class-validator DTOs as the older way), **guards**, **interceptors** and **exception filters**

WEEK	FOCUS AREA	TOPICS
		- Config module and **OpenAPI/Swagger** docs; how a **React** or **Next.js** page calls your API, **CORS** and a typed client at awareness - **Authentication**: password hashing with **argon2**/**bcrypt**, **sessions vs JWT**, refresh-token rotation and revocation, built as a real Nest guard - The week's build: your Express and Prisma API rebuilt in NestJS, with register and login
Week 7	Query Plans, Transactions, Redis, Background Jobs & Docker	- Reading query plans with `EXPLAIN ANALYZE` line by line, the **N+1** trap, Prisma query logging and connection pools; PgBouncer at awareness - **Transactions** and ACID (all steps of a write happen or none do), Postgres's default **isolation level**, and **locking**: lost updates, `SELECT ... FOR UPDATE`, optimistic locking and a clean `409` on a unique conflict; deadlocks at awareness - **Redis**: data structures, cache-aside, TTL, **invalidation** and cache stampedes, then rate limiting with a fixed-window counter; sessions in Redis and distributed locks at awareness - Background and scheduled jobs with **BullMQ** and its **Job Schedulers**: retries with backoff and jitter, dead-letter queues, idempotent consumers and **idempotency keys** for safe retries; a work queue vs pub/sub, with **Kafka** and **RabbitMQ** named; the outbox at awareness - **Docker**: Dockerfiles, multi-stage builds, non-root containers, **Docker Compose** with app, PostgreSQL and Redis, and graceful shutdown

Phase 3 · Security, AI Backends & System Design (Weeks 8–9, 20 Hours)

WEEK	FOCUS AREA	TOPICS
Week 8	Account Flows, Permissions, Security, Webhooks, Real-Time & AI in the Backend	- Account flows and email: **email verification** and **forgot password** with a single-use token, sent from a BullMQ job through a free tier or **Mailpit** - **Authorisation**: **RBAC** and object-level checks as NestJS guards; **OAuth2**/**OpenID Connect** "login with Google" as a mentor demo - **OWASP API Security Top 10 (2023)**, BOLA first, **Helmet**, CORS, rate limiting, secrets, file uploads with presigned URLs to the Cloudflare R2 free tier or a local container

WEEK	FOCUS AREA	TOPICS
		- **Webhooks**: checking the signature, answering fast and queuing the work, and handling the same event twice safely - Real-time: **WebSockets** with NestJS gateways and Server-Sent Events; the Redis socket adapter at awareness - **AI in the backend**: an **LLM API** call on a free tier such as **Gemini** or **Groq** (structured output, streaming, timeout, cost per request, a checked answer)
Week 9	AI Backends, Scaling & System Design	- **Tool calling** and a simple agent loop: the model asks your backend to run two or three of your own functions, with a step limit and the user's own permissions; **MCP** (Model Context Protocol) at awareness - **Embeddings** and search by meaning (finding text that means the same thing, not just the same words), stored and searched in PostgreSQL with **pgvector** - **RAG** (retrieval-augmented generation): answering from your own documents, split into chunks, retrieved for the user who owns them, and answered with sources - **Guardrails, evals and cost**: prompt injection, keeping private data out of prompts, a small test set that scores answers, per-user limits and caching to control spend - **Scaling and designing real systems**: stateless servers behind a **load balancer**, caches and a **CDN**, **read replicas** and replication lag; **sharding** at awareness; monolith vs microservices, **Kubernetes** and **Terraform** named; then a **payments** flow designed together in the interview format (retries that cannot charge twice, and your server calling **eSewa** or **Khalti** to verify a payment), including where an AI feature sits so the app keeps working when the model is down; **circuit breakers** (closed, open, half-open), the token bucket most production rate limiters use, and CAP at awareness; the design doc is written at home

Phase 4 · Production Deploy & Career (Week 10, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 10	Production Deploy & Getting Hired	- Your Compose stack on the batch server: **Nginx** reverse proxy, **Let's Encrypt** HTTPS on your subdomain, and a mentor walkthrough of server hardening (SSH keys, firewall, updates); then the mentor maps your stack to **AWS** (EC2, S3, RDS, ElastiCache, SQS, CloudWatch)

WEEK	FOCUS AREA	TOPICS
		- **CI/CD** with **GitHub Actions**: lint, type-check, build, image push, deploy on `main`, rollback - Logs, `/health` and `/ready`; `pg_dump` backups with a practised restore - **Day 3 · CV, job applications and LinkedIn:** an ATS-friendly backend CV built around your projects with real numbers, where backend jobs are posted (company career pages, **merojob**, **KumariJob**, **JobAxle**, **LinkedIn**) and remote roles, reading a job post, a target-company list and tracker, **GitHub** and an improved **LinkedIn** profile - **Day 4 · Mock interview:** one mock interview for every student (API design, SQL, a short system-design question, a **read-and-explain round with AI tools off**, and a **timed live-debugging drill, AI allowed**), completed with written feedback, no pass or fail - **Day 5 · Project showcase and presentation:** your projects presented in 6 minutes each with a live demo and a recorded backup, then questions and feedback

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Web & Protocols	HTTP, DNS, TCP, TLS/HTTPS, REST; HTTP/2, HTTP/3, GraphQL and gRPC awareness
Language & Runtime	JavaScript, TypeScript (generics, strict mode), Node.js (event loop, async/await), calling outside APIs, reading and tracing code, Linux terminal, Git
API Design	Express (v5), Zod validation, error contracts, versioning, cursor pagination, idempotency keys, OpenAPI/Swagger, Bruno, pull requests and code review
NestJS & Architecture	NestJS (v12): modules, dependency injection, guards, pipes, interceptors; SOLID, design patterns
Databases	PostgreSQL, SQL (JOINs, CTEs, window functions), data modeling, Prisma 7, indexes, EXPLAIN ANALYZE, transactions, isolation levels, locking; MongoDB awareness
Caching & Async	Redis (caching, rate limiting, and how sessions are kept in Redis), BullMQ background jobs and Job Schedulers, webhooks, WebSockets

CATEGORY	TECHNOLOGIES
Auth & Security	argon2/bcrypt, sessions, JWT, email verification and password reset, OAuth2/OpenID Connect, RBAC, OWASP API Security Top 10, Helmet, rate limiting
AI Backends	LLM API calls (structured output, streaming, timeouts, cost), tool calling, MCP awareness, embeddings, pgvector, RAG, prompt-injection guardrails, evals, AI cost control
System Design	Load balancing, caching/CDNs, read replicas, sharding awareness, estimation, timeouts and retries, circuit breakers and token bucket awareness, design docs
DevOps	Docker, Docker Compose, Nginx, Let's Encrypt, GitHub Actions CI/CD, health checks, backups
AI Productivity	Cursor, GitHub Copilot (clear tasks, asking why, checking every line)

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 and TypeScript core (values, functions, arrays, objects, JSON, interfaces, unions)	Depth, one full week
Node.js, async code, closures, classes and calling an outside API	Depth, one full week
Reading, tracing and explaining code, including code an AI wrote	Depth, drilled all course
Reading errors and debugging	Depth, drilled all course

FOCUS AREA	LEARNING FOCUS
Express and REST API design (validation, errors, pagination) and a first deploy	Depth
SQL, PostgreSQL, data modeling and Prisma 7	Depth, one full week
Indexes and EXPLAIN	Depth
NestJS 12 (modules, DI, pipes, guards, interceptors), design patterns and login	Depth, one full week
Transactions, isolation and locking, and Redis	Depth
Background and scheduled jobs (BullMQ, idempotent consumers, idempotency keys)	Hands-on
Docker and Docker Compose	Depth
Authentication, authorisation and OWASP API Security Top 10	Depth
Email verification and forgot password	Hands-on
"Login with Google" (OAuth2/OpenID Connect)	Mentor demo
Webhooks (signatures, repeats)	Hands-on
WebSockets and real-time	Hands-on
LLM API calls (structured output, streaming, timeout, cost, checked answers)	Hands-on, guided
Tool calling and a simple agent loop	Hands-on, guided
Embeddings, pgvector and RAG over your own documents	Hands-on, one day each
AI guardrails, small test sets and cost control	Hands-on
Scaling from one server to many (load balancers, caches, read replicas)	Concept
Timeouts and retries, and designing a payments system live in the interview format	Depth, one live design; the design doc written at home

FOCUS AREA	LEARNING FOCUS
Deploy with Nginx/HTTPS, GitHub Actions CI/CD, logs, health checks and backups	Depth
AWS core, matched to what you already run	Awareness, a mentor demo
Kafka, RabbitMQ, Kubernetes, Terraform, PgBouncer	Named only
TypeORM, GraphQL, gRPC, MySQL, MongoDB, sharding, CAP, MCP, circuit breakers, rate-limiter algorithms	Awareness

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
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 with AI 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. You do not need to pay for a server: the academy gives each student a login and subdomain on its batch server.

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 prepares you for junior backend roles, with the database and system design depth mid-level interviews ask, and the AI backend skills (tool calling, RAG, guardrails) that newer job posts ask for. Tools for much larger systems, such as Kubernetes, are only named. 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. You get a full week on JavaScript and TypeScript, then a full week on Node.js and async code.
- You read and run code before you write it, so you understand each piece before you build with it.
- 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). Your mentor gives you your projects during the course, and you put your work online from Week 4. In the final week you do a mock job interview.

Which stacks and tools will I actually use?

- **Node.js**, **TypeScript**, **Express**, **NestJS 12**
- **PostgreSQL**, **Prisma**, **Redis**, **BullMQ**, webhooks, WebSockets, JWT, OAuth2
- An **LLM API** on a free tier such as **Gemini** or **Groq**, with tool calling, and **pgvector** for search by meaning and RAG
- **Docker**, **Docker Compose**, **GitHub Actions**, Nginx/HTTPS
- **MongoDB** and **AWS** at awareness level; **Kubernetes** and **Terraform** named only

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, AI features, system design and deployment.

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 often start with a technical test, written at their office at some companies, 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.

Some companies still set a basic written test, with a few lines of code on strings and arrays and questions on SQL joins and normalisation (organising database tables). We do not drill puzzles. You spend ten weeks reading, tracing and explaining code, which is what these tests check, and SQL is covered in Weeks 5 and 7. Some interviews allow AI tools and some ban them, so the Week 10 mock interview on Day 4 has one round with AI tools off and one with them on.

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.
- 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, or live online. 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, 9768618434
- Limited to 10 seats per batch

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

Version 1.2.0 · Updated 6 Mangsir 2083