TR-12

Backend Engineering

COURSE BRIEF

SKILLS & TOOLS YOU WILL MASTER

Node.js + Express | NestJS + TypeScript | System Design | PostgreSQL + Redis | Docker + CI/CD

10 Reasons This Works.

Why students choose Saarathi - and how each card helps you ship a real career.

01

10 Max / Batch

Not 50. Not 30. Ten. Every student seen, heard, and unblocked - every class.

02

Sunday Open Classroom

Every Sunday the classroom is open. Come in, pair-program, get unstuck, or just focus.

03

Weekly Detailed Syllabus

Full syllabus from day one. Every week: what to read, what to build, what the outcome is.

04

Revision Every 5th Day

Every fifth class is a revision session. Nothing piles up or slips through.

05

Week 1: Mindset

The first week is about getting into the right headspace before touching a line of code.

06

3 Ways of Learning

Self-learning + collaboration + mentorship. All three built into every course.

07

1:1 Mock Interview

Real mock interview with your mentor in the final week. Actual hiring prep, not fluff.

08

CV + LinkedIn

Resume writing and LinkedIn optimisation. Leave with a profile that gets replies.

09

Real Projects

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

10

Gate Assessment

Aptitude test before Day 1. Maps your strengths, weaknesses, learning style, and pace - then builds your personal plan.

Backend Engineering

Why This Course

This course is for people who want to become junior backend developers, the engineers who build the servers, APIs and databases behind every app. The frontend is what a user sees; the backend is everything behind it. Employers in Kathmandu and on remote teams hire for exactly this: someone who can design a clean REST API, model data in PostgreSQL, secure it with real authentication, and ship it running on a server. Pay scales with your skills, portfolio, and whether you work locally or for remote clients; we don't promise a salary.

You learn **one backend mental model**, request → route → validate → business logic → database → response, and drill it deep in the **Node.js** stack: first in **Express** (the thinnest, most in-demand Node framework and the strongest first-job signal in Nepal), then in **NestJS** (structured and TypeScript-first, how larger teams organise a backend). You then spend a full week on **system design** learning how a backend scales, and take your best backend all the way to production with Docker and a live deployment. Depth in one stack, then shipped for real, beats a shallow tour of many.

Two things worth knowing before you enrol, because nobody tells beginners either of them. Kathmandu also hires a lot of PHP and Laravel, so Node.js roles are a real but smaller slice of the local market, and the place this skill set pays best is remote and international teams. And most local listings tagged "entry level" still ask for a year. That is why this course ends with a deployed, defensible project instead of a certificate: a live URL and a repo you can explain line by line is what stands in for the experience line on your CV.

It is foundations-first. The first four weeks, forty percent of the course, have no framework at all: how the web works (HTTP, DNS, TLS, REST), the terminal and Git, JavaScript from absolute zero across two full weeks (the basics, then asynchronous code and the Node.js runtime), and SQL with PostgreSQL and data modeling. The language gets two weeks and the database gets its own because that is where beginners actually stall. Only then do you touch a framework. AI pair-coding (**Cursor**, **GitHub Copilot**) is woven in with strict verify-discipline: you read and understand every generated line.

Who Is This Course For

- **Complete beginners** who want a real backend engineering career and will do the foundations work
- **Frontend or JavaScript learners** moving to the server side to become full stack
- **Students and career changers** targeting a first junior backend role in Nepal or remote

- **Self-taught coders** who can script but have never designed an API, a schema or an auth system
- **Anyone** who wants to go deep in the Node.js backend stack, learn system design, and ship it to production

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 35,500
Level	Beginner-friendly, taught from the fundamentals → Junior Backend Developer
Outcome	Junior Backend Developer (Node.js/Express + NestJS, system design, deployed to production), job-ready for Nepal and remote teams

Prerequisites

- No prior coding required, JavaScript is taught from scratch across Weeks 2 and 3
- A laptop with 8GB+ RAM, any modern Windows, macOS or Linux machine, no special hardware
- Comfortable using a computer daily, files and folders
- Basic English reading and writing, you will write READMEs, commit messages and progress updates, and clear written English is what gets Nepal-based engineers onto remote international teams
- Saarathi Gate Assessment before Day 1, diagnostic, no pass or fail

Nothing to install beforehand. VS Code, Node.js, Git, GitHub and the free Postgres host are set up together in the first week.

Assessments at a glance

Two exams bracket the course. The **Saarathi Gate Assessment** runs before Day 1 as a diagnostic with no pass or fail; it maps your starting point and pace. The **Saarathi Certification Exam** is sat at the end, 2 hours and around 30 questions specific to this course in mixed formats (single choice, multi-select, true/false, ordering, matching, short answer, and code and short case studies where relevant). Pass mark is 60 percent and a pass is mandatory for the Saarathi Certificate of Completion. The exam is camera proctored: a working webcam must stay on for the full two hours, and stills are saved for your trainer to review. You can answer the questions in any order and come back to any of them, but every question needs an answer before you submit. Recommendation letters are discretionary, mentor-issued only to learners who pass, and never guaranteed.

Your Learning Week

DAY	ACTIVITY
Mon–Fri	2-hour live class session (hands-on, project-driven) with mentor
Mon–Fri	2-hour self-study at home (building endpoints, debugging, reading docs)
Sunday	Open classroom for focused build time or rest
Day 5 (every week)	Within the 2-hour session, the last 1 hour is a built-in revision block plus architecture review for consolidation and doubt-clearing

> Real improvement happens when you build, break, debug and re-ship between classes. The classroom gives structure; the hours between classes give depth.

How the Difficulty Is Paced

Every week carries **at most seven topics**, and no week asks you to learn two brand-new hard things at once.

- **Weeks 1–4 are the ramp.** One new idea at a time. JavaScript gets two full weeks (the basics, then asynchronous code and the Node.js runtime) because the language is the biggest beginner wall in any backend course, and SQL gets a week of its own.
- **Weeks 5–7 are the climb.** Every framework concept arrives as a name for something you already built by hand: Express is your own Week-3 server with the boilerplate removed, NestJS is your own Week-5 structure made official.

- **Week 8 is applied production work, Week 9 is a thinking week.** System design has nothing to install and nothing to debug, so it is also the breather before the final push.
- **Week 10 is delivery.** No new fundamentals, just shipping what you built and turning it into a job application.

Every week states what you should be able to do by the end of it and what to build between classes. Batches are capped at ten so your mentor tracks those checkpoints per student.

Week-by-Week Curriculum

Phase 1 · Web, JavaScript, Node.js & SQL Foundations (Weeks 1–4, 40 Hours)

WEEK	FOCUS AREA	TOPICS
Week 1	How the Web Works, the 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 - **HTTP** in depth: methods, status codes, headers, body, inspected with dev-tools and `curl` - **REST**: resources as URLs, verbs as actions, statelessness - Your machine set up once: **VS Code**, **Node.js**, **Git**, **GitHub**, a free **PostgreSQL** tier, plus the **Cursor/Copilot** verify-discipline rule set - Terminal and shell basics: `ls` / `cd` / `cp` / `mv` / `rm`, paths, pipes - **Git** and **GitHub** from zero: commit, push, branch, the pull-request workflow
Week 2	JavaScript Foundations	- Values, variables (`let` / `const`), types, template literals, operators and truthiness - Control flow: `if` / `else` and loops (`for` , `while` , `for...of`) - **Functions**: parameters, defaults, return values, scope, arrow functions - Arrays and passing functions around: `map` / `filter` / `reduce` - Objects, destructuring and spread, and **JSON**, the format every API speaks - Reading errors and debugging: stack traces, the classic beginner errors, `console.log` vs the VS Code debugger - Small programs, each committed to Git as you go
Week 3	Async JavaScript, the Node.js	- Error handling: `try` / `catch` , throwing, `Error` objects, why silent failures kill backends - Callbacks and why async exists: the server cannot freeze while it waits

WEEK	FOCUS AREA	TOPICS
	Runtime & Your First Server	- **Promises**: pending, fulfilled, rejected, `.then` / `.catch` and chaining - `async` / `await` : the readable form, error handling, sequence vs `Promise.all` - **Node.js** as a runtime: the event loop at intuition level, built-in modules, ES modules vs CommonJS - **npm**, `package.json` , packages, `.env` with **dotenv** - Your first HTTP server with Node's built-in `http` module (before the framework)
Week 4	SQL, PostgreSQL & Data Modeling	- Why a database, and **relational vs NoSQL** (**MongoDB** awareness), chosen by access pattern - **PostgreSQL** setup, the `psql` client and a GUI client - Core **SQL**: `CREATE` / `INSERT` / `SELECT` / `WHERE` / `UPDATE` / `DELETE` , the CRUD verbs behind endpoints - Relationships and **JOINs**: primary/foreign keys, one-to-many, many-to-many, aggregates - **Data modeling** and **normalisation**, with an entity-relationship diagram before code - **Constraints** (`NOT NULL` , `UNIQUE` , `CHECK`) and **transactions** (`COMMIT` / `ROLLBACK`) - SQL from Node with the `pg` driver and parameterised queries (the SQL-injection cure)

Phase 2 · Express REST APIs, Auth & Security (Weeks 5–6, 20 Hours)

WEEK	FOCUS AREA	TOPICS
Week 5	Express Fundamentals - Routing, Middleware, Data & Errors	- **Express** (v5, current default on npm): the app object, `Router` , route params, query strings - The **middleware** pipeline: ordering, `next()` , body parsing, `cors` , why it is Express's core idea - Layered architecture: routes, controllers and a service/data layer - Your database behind the API: a `pg` connection pool, parameterised queries per endpoint, database errors handled safely - Request validation with **Zod** (never trust the client) and clear `400` errors - Central **error handling**: custom errors, consistent JSON shapes, correct status codes

WEEK	FOCUS AREA	TOPICS
		- Twelve-factor config, git-ignored `.env` + `.env.example`, structured logging
Week 6	Authentication, Authorisation & API Security	- **Auth** concepts: authentication vs authorisation, password hashing with **bcrypt** - **Sessions vs JWT** (JSON Web Tokens): how each works, tradeoffs, refresh rotation - Protecting routes: auth middleware, **RBAC** and object-level checks - **OWASP API Security Top 10**: **BOLA**, broken auth, excessive data exposure, injection, misconfiguration, with the fix for each - Hardening: **Helmet**, CORS, **rate limiting**, request-size limits - **OAuth2** and "login with Google" at working level - File uploads with **Multer** (`multipart/form-data` , type/size checks) as an auth-sensitive surface

Phase 3 · NestJS, Prisma & Testing (Week 7, 10 Hours)

WEEK	FOCUS AREA	TOPICS
Week 7	TypeScript, NestJS, Prisma & Testing	- Just-enough **TypeScript** (types, interfaces, decorators) and why **NestJS** (v11, current) is TypeScript-first - **Modules**, **controllers** and **providers/services**, the layered split made first-class; monolith vs **microservices** named - **Dependency injection**: what it is and why it makes code testable (intuition first) - **Pipes** (validation), **guards** (your JWT logic reused) and **interceptors**, each mapped back to Express middleware - **Prisma** ORM (the recommended TS ORM): schema, migrations, the type-safe client, relations, transactions; **TypeORM** as the alternative (awareness) - DTOs as typed request contracts, config module, exception filters, **OpenAPI/Swagger** docs - Automated testing: **Jest** + **Supertest** against your Express API, then unit and end-to-end tests the Nest way

Phase 4 · Production Layers & System Design (Weeks 8–9, 20 Hours)

WEEK	FOCUS AREA	TOPICS
Week 8	Performance, Caching, Real-Time & Running the Whole Stack	- Query performance: **indexes**, reading `EXPLAIN`, connection pooling, profiled on your real API - Why caching, and **Redis**: in-memory store, `GET` / `SET`, TTL, caching a slow endpoint - Cache strategy and the hard part, **invalidation**; where caches live (client, CDN, app, database) - **WebSockets** and real-time with **Socket.IO** / Nest gateways, mapped alongside REST - Background jobs and **message queues**: producer/consumer with **BullMQ** on Redis, retries and idempotency (Kafka/RabbitMQ awareness) - When to reach for each: index, cache, queue, WebSocket, or a plain synchronous endpoint - **Docker** multi-stage build + **Docker Compose** (app + PostgreSQL + Redis, volumes, healthchecks) so the stack starts with one command
Week 9	System Design - How a Backend Scales	- What system design is: latency vs throughput, single points of failure, **back-of-the-envelope** estimation (QPS, storage, bandwidth) - **Scaling** vertical vs horizontal, stateless services, a **load balancer** and health checks/failover - **Caching & CDNs** at scale: where caches live across a whole system, and invalidation once several caches hold the same answer - **Database scaling**: read **replicas**, **sharding/partitioning**, indexing, connection pooling, SQL vs NoSQL by access pattern - **Async at scale**: message queues, producer/consumer and event-driven, **idempotency**, eventual consistency - Reliability and the **CAP** tradeoff: consistency vs availability, timeouts, retries, rate limiting, graceful degradation - **Design a real system** end to end (URL shortener / news feed / chat) in the system-design interview format

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

WEEK	FOCUS AREA	TOPICS
Week 10	Ship to Production, Deliver the Capstone & Get Hired	- Deploy to a real **VPS**: SSH, hardened non-root user, firewall, running your Week-8 Compose stack off your laptop - **Nginx** reverse proxy + **Let's Encrypt** HTTPS, a public `https://` URL - **CI/CD** with **GitHub Actions**: lint, test, build on every push, deploy on `main`, versioned rollback - Config, secrets, safe migrations and a scheduled **PostgreSQL** backup (`pg_dump`) with a practised restore - **Observability**: structured logging, request IDs, `/health` endpoints, uptime monitoring, reading live logs - Capstone sign-off, a recorded demo and a portfolio-grade repo (architecture diagram, data model, API docs, coverage figure) - Resume, **GitHub** and LinkedIn, a mock interview (API design + SQL/data modeling + capstone defence), target-company list and outreach cadence

Skills You Will Gain

CATEGORY	TECHNOLOGIES
Web & Protocols	HTTP methods/status codes, DNS, TCP, TLS/HTTPS, REST API design
Language & Runtime	JavaScript (async/await, JSON, error handling, debugging), Node.js, TypeScript (for NestJS)
Databases	PostgreSQL, SQL (JOINs, aggregates, constraints, indexes, transactions), data modeling and normalisation, MongoDB awareness
Node.js Frameworks	Express (v5): routing, middleware, Zod validation, error handling, Multer uploads; NestJS (v11): modules, DI, guards, pipes, interceptors
ORMs	Prisma (recommended TS ORM), the `pg` driver; TypeORM awareness
Auth & Security	bcrypt, sessions, JWT, OAuth2, RBAC/object-level authorisation, OWASP API Security Top 10, Helmet, CORS, rate limiting

CATEGORY	TECHNOLOGIES
Testing & Docs	Jest, Supertest, unit + end-to-end tests, OpenAPI/Swagger
Production Layers	Query tuning with indexes and EXPLAIN, Redis caching + invalidation, WebSockets (Socket.IO / Nest gateways), BullMQ queues; Kafka/RabbitMQ awareness
System Design	Scaling & load balancing, caching/CDNs, database scaling (replicas, sharding, indexing), queues & async, CAP/consistency tradeoffs, back-of-envelope estimation, end-to-end system design
Deployment & DevOps	Docker, Docker Compose, VPS deploy, Nginx reverse proxy, Let's Encrypt HTTPS, GitHub Actions CI/CD, database backups, observability (logging, health, uptime)
AI Productivity	Cursor, GitHub Copilot (with verify-discipline)

Topic Depth and Awareness

> **Depth** means repeated practice until you can apply the concept confidently. **Awareness** means you are introduced clearly enough to read and compare it later.

FOCUS AREA	LEARNING FOCUS
Web fundamentals (HTTP, DNS, TLS, REST, status codes)	Depth
Terminal and Git (branching, pull requests)	Depth
JavaScript basics (values, control flow, functions, arrays, objects, JSON)	Depth, one full week
Async JavaScript (callbacks, Promises, async/await) and the Node.js runtime	Depth, one full week
Reading errors and debugging	Depth, drilled all course
SQL and PostgreSQL (JOINs, constraints, transactions, parameterised queries)	Depth
Data modeling and normalisation	Depth

FOCUS AREA	LEARNING FOCUS
Express (routing, middleware, database layer, validation, error handling, uploads)	Depth
Authentication and authorisation (bcrypt, sessions, JWT, RBAC)	Depth
OWASP API Security Top 10 (BOLA, hardening)	Depth
NestJS (modules, DI, guards, pipes) and Prisma	Depth
Automated testing (Jest, Supertest) and API docs (OpenAPI/Swagger)	Depth
Query performance (indexes, EXPLAIN) and Redis caching with invalidation	Depth
Docker, Docker Compose, VPS deployment, Nginx/HTTPS, GitHub Actions CI/CD, backups, observability	Depth
System design: scaling, load balancing, caching/CDN, database scaling, queues, CAP tradeoffs	Depth (concept), one full week
Designing a real system end to end (interview format)	Depth, one guided design
OAuth2 / third-party login	Working level, one guided walkthrough
WebSockets and real-time	Hands-on intro
Message queues (BullMQ)	Hands-on intro
Kafka / RabbitMQ	Awareness
MongoDB and NoSQL tradeoffs	Awareness
TypeORM (vs Prisma)	Awareness
Microservices architecture	Awareness
Managed hosts (Railway, Render, Fly.io), cloud databases, Kubernetes	Awareness

Project Pool

Each student picks **one** capstone with their mentor in the Week 5 architecture-review slot, then builds, secures, tests, layers and deploys it across Weeks 5–10. Pool rotates per batch:

#	PROJECT	STACK FOCUS
1	Task/project management API with users, teams and role-based access	Express or NestJS, JWT, RBAC, PostgreSQL
2	E-commerce backend: products, cart, orders, payments-stub, inventory	NestJS + Prisma, transactions, Redis cache
3	Booking/reservation system with availability and conflict handling	Express, transactions, validation, WebSockets
4	Content/blog platform API with auth, media uploads and search	Express or NestJS, file uploads, full-text search
5	Real-time chat or notifications service	NestJS gateways / Socket.IO, Redis, queues
6	URL shortener or analytics API with caching and rate limiting	Express or NestJS, Redis, indexing

Career Paths & Trajectory

ROLE	TIMELINE	FOCUS
Junior Backend Developer / Backend Intern	First 3–6 months	Build and fix API endpoints under review; convert the portfolio into a first junior role
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 35,500. 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 prior coding is required. JavaScript is taught from scratch across Weeks 2 and 3, and the first four weeks are pure foundations with no framework at all. You do need computer fluency, comfort following setup guides, and a laptop. The Saarathi Gate Assessment runs before Day 1 to map your starting point, diagnostic, not pass-or-fail.

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

The pacing is built for exactly that. No week carries more than seven topics, JavaScript gets two full weeks instead of the usual one, SQL gets a week of its own, and every week ends with a checkpoint saying what you should now be able to do plus a set list of what to practise between classes. If something has not clicked, the Day-5 revision hour, the Sunday open classroom and a batch capped at ten students are there for it.

How long is the course?

10 Weeks (2.5 Months). The capstone is scoped in Week 5 and built inside the course, all the way to a live deployment and a mock interview in the final week.

Which stacks and tools will I actually use?

Depth in **Node.js with Express and NestJS**, plus **Prisma** and **PostgreSQL**, JWT auth, Redis, WebSockets and queues. Then a full week of **system design**, and production deployment with Docker, Docker Compose, Nginx/HTTPS, a GitHub Actions CI/CD pipeline and observability. Every phase ships a real, reviewable artefact, no toy projects.

Why one stack instead of many?

Because a shallow tour of five frameworks teaches none of them. You share one backend mental model, request → route → validate → logic → database → response, and go deep in Node.js with Express then NestJS, so it becomes second nature. The saved time buys four unhurried foundation weeks, a full system-design week and a real production deployment, the things that actually separate a hireable junior from a tutorial-follower.

What does a backend interview in Nepal actually look like?

It depends on the size of the company, and we train for all three. Large product companies run an online coding assessment, then a technical interview, then a second technical or design round. Mid-sized companies, which are most of the market, set a written technical test on site

first, one to two hours, then interview you. Startups run an informal chat with the founder or lead, then a take-home task over one to three days. The written round is more basic than people expect, reversing a string, finding the largest number in an array, checking a palindrome, all by hand, plus SQL joins and normalisation. We drill those in the Week 2 and Week 4 revision hours, and run the full mock loop in Week 10.

What roles does this prepare me for?

Roles are listed in the Career Paths section, starting with Junior Backend Developer. We do not promise jobs. We build your portfolio, run mock interviews, polish your CV, GitHub and LinkedIn, and refer you into our hiring network. Be clear-eyed about the split: local junior backend pay is modest wherever you land, and remote and international teams are where these skills change your earnings the most. Those teams gate on two things you can control, a portfolio somebody can open and run, and clear written English in async updates. Both are built into the course.

How does the batch and mentorship work?

Maximum 10 students per batch, so every student gets direct mentor time and project review. Hybrid by default, in-person at Old Baneshwor with online recordings and office hours. Sunday Open Classroom is free for enrolled students: pair-program, get unstuck, or just focus.

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 2.0 · Updated August 11, 2026