IMPACTEERS

DEPLOYMENT ENGINEERING CERTIFICATION · COHORT 01

The AI Forward Deployed Engineer Program

Data Integration · Enterprise Deployment · AI on Client Data · Client Engagement

A six-month, weekend-live programme for engineers who deliver software inside organisations they do not control — extracting from systems nobody documented, deploying into environments nobody owns, and handing over so completely that the client no longer needs them.

24	20	40	7	1
CALENDAR WEEKS	SPRINTS	PROJECTS	MILESTONES	LIVE CLIENT ENGAGEMENT

144	96	240	W19
LIVE HOURS (SAT & SUN)	SELF-PACED HOURS	TOTAL STRUCTURED HOURS	CLIENT ASSIGNED

Entry requirement

This is not an entry-level programme. Admission requires either completion of the Impacteers AI-Native Full-Stack Engineer Program, or two or more years of professional software engineering experience. Forward deployed roles put engineers in front of client executives from week one of an engagement; the market does not hire freshers into them, and this programme does not pretend otherwise.

Contents

1. What a Forward Deployed Engineer Actually Does

A product engineer builds software for users they will never meet, inside a codebase and infrastructure their company controls. A forward deployed engineer does almost the opposite: they sit inside a client organisation, work out what the problem actually is, extract data from systems nobody documented, deploy into infrastructure they were granted access to last Tuesday, and leave behind something the client's own team can run.

The engineering skills overlap. The job does not. This programme is built around the four things that make the role distinct.

The distinction	What it means for this programme
You do not own the data	Client data lives in a 2009 Oracle instance, an SFTP drop, a spreadsheet maintained by one person in accounts, and an API with no versioning. Four weeks of this programme are spent on extraction, reconciliation and data quality — because that is where the hours actually go.
You do not own the environment	Deployment means their cloud account, their Kubernetes version, their identity provider, their firewall rules, their change window and their security questionnaire. Kubernetes, Terraform, Vault and enterprise SSO are taught at implementation depth, not as awareness topics.
The requirement does not exist yet	Clients describe symptoms, not specifications. Discovery, scoping, estimating and scope defence are assessed skills here, with a full sprint devoted to turning a vague complaint into a signed statement of work.
You are measured on becoming unnecessary	The engagement succeeds when the client's team can operate, extend and troubleshoot the system alone. Runbooks, enablement sessions and handover rehearsals are graded deliverables, not afterthoughts.

What this programme claims

- Associate forward deployed engineering readiness: full-stack capability plus data integration, enterprise deployment and client engagement.
- Working fluency with the integration stack — Python, Airbyte, Temporal, Great Expectations — against undocumented source systems.
- Deployment capability into client-controlled environments: Kubernetes, Helm, Terraform, Vault, enterprise SSO, air-gapped delivery.
- AI deployment on client data with permission-aware retrieval, evaluation on a client-signed golden set, and an honest business case.
- A defensible engagement portfolio: discovery brief, scoped SOW, deployed system, runbook, handover pack and a defended capstone.

What this programme does not claim

- Guaranteed employment. Career claims must match demonstrated evidence.
- Senior or lead FDE readiness. The honest outcome is a credible associate who can be deployed alongside an experienced engineer.
- Model training, fine-tuning or foundation-model research capability. AI here means deployment, retrieval and evaluation.
- Compliance certification. Learners produce evidence and map controls; they do not become auditors.
- Real client access. Engagements are simulated with role-playing stakeholders, which is stated openly to employers.

2. The Learning Model

The programme runs on a strict two-layer split. Weekdays are for acquiring language, syntax and tool surface at the learner's own pace on the Impacteers platform. Weekends are for the things that only work with an engineer in the room: implementation, architecture argument, live debugging, client conversation and review.

The split contract

	Self-paced · Mon–Fri · ~4 hrs	Live studio · Sat & Sun · 3 hrs each
Owns	– Concept micro-videos and reading notes – Language syntax and tool configuration surface – Guided setup: clusters, providers, connectors – Browser-IDE and sandbox coding practice – Weekly quiz and debugging challenge – Technical problem sets with complexity notes – Cheat sheets and pre-session readiness check	– Implementation of the sprint's projects, built live – Architecture decisions argued in the open – Failure drills: broken pods, changed schemas, dead sources – Client conversations with role-playing stakeholders – Code review, pull requests and demos – Field Labs — AI-native or client engagement – Technical Clinic: patterns, system design, case practice
Never	– Sole exposure to a topic that will be assessed live – Unverified AI-generated explanation	– Teaching syntax from scratch – Passive lecture with no artifact produced – Sessions that end without working code or a written deliverable
Format	Video, notes, interactive coding, quizzes, problem sets — all on the Impacteers platform, available throughout the programme.	Instructor-led, camera-on, screen-shared. Client role-plays are recorded and critiqued. Sessions published within 24 hours.

The readiness rule

Live sessions do not teach syntax. Learners who arrive without the week's self-paced module complete may attend and participate, but do not receive the readiness credit that contributes to the Live Studio Participation component of the final grade. This matters more here than in a standard engineering programme: a learner who has not configured Terraform before Saturday cannot participate in a live provisioning session, and a learner who has not read the discovery material cannot run a stakeholder interview.

Time commitment

Component	Per week	Notes
Self-paced platform learning	~4 hours	Mon–Fri, flexible. Concept videos 1.25 hrs, interactive practice 1 hr, guided lab 1 hr, quiz and debugging 0.5 hrs, revision and readiness check 0.25 hrs.
Saturday Implementation Studio	3 hours	Live. The sprint's Applied Build project constructed in-session, with failure drills.
Sunday Field Engineering Studio	3 hours	Live. Technical Clinic, sprint project, Field Lab and sprint review.
Independent project work	4–6 hours	Completing the sprint project, tests and evidence between sessions.

Component	Per week	Notes
Placement track (from Week 13)	~3 hours	Applications, mock interviews, recruiter conversations, profile work.
Client engagement (from Week 19)	~3 hours	Discovery, stakeholder correspondence, build and handover for the assigned capstone client.
Total	11–16 hours	Rising to 17–20 hours from Calendar Week 19 when the client engagement begins.

3. The Weekly Rhythm

Every week follows the same shape. The predictability is deliberate: learners plan around it, and instructors reuse the structure so studio time is spent on engineering and client work rather than logistics.

Monday to Friday · Self-paced foundation

Block	Time	What the learner does
Concept videos and reading	1 hr 15	Micro-videos capped at five minutes each, paired with written notes. Covers language rules, syntax and tool configuration surface for the sprint.
Interactive practice	1 hr 00	Browser IDE and sandbox drills. Fluency work — typing the syntax and the CLI until it stops costing attention.
Guided lab	1 hr 00	A structured hands-on task with setup, steps, acceptance criteria and a submission checklist. Prepares the ground for Saturday.
Quiz and debugging challenge	0 hr 30	Knowledge check plus a broken configuration, query or pipeline to diagnose. Contributes to the grade.
Technical problem set	ongoing	Five problems per week from Calendar Week 3, each submitted with a written complexity justification. Shifts to system design and case prep from Week 13.
Revision and readiness check	0 hr 15	Cheat sheet plus a short readiness quiz that unlocks the weekend credit.

Saturday · Implementation Studio · 3 hours live

Clock	Segment	Purpose
0:00 – 0:20	Readiness check and blocker clinic	Rapid confirmation that the self-paced layer landed. Blockers surfaced and cleared before building starts.
0:20 – 1:40	Live implementation walkthrough	The instructor builds the sprint's Applied Build project from empty, narrating every decision. Learners follow in their own repository or environment.
1:40 – 1:50	Break	
1:50 – 2:40	Learner build sprint	Learners extend the build independently or in pairs while instructors circulate. This is where the work becomes theirs.
2:40 – 3:00	Failure drill and brief	The instructor breaks something deliberately — a pod, a source schema, an external service — and the group diagnoses it. Sunday's scope briefed.

Sunday · Field Engineering Studio · 3 hours live

Clock	Segment	Purpose
0:00 – 0:45	Technical Clinic	Weeks 3–12: DSA pattern teaching and one timed problem under

Clock	Segment	Purpose
		interview conditions. Weeks 13–24: system design and case-interview practice, which is what forward deployed loops actually assess.
0:45 – 1:45	Sprint project implementation	The second project of the sprint built live — the one that extends Saturday's work rather than starting over.
1:45 – 1:55	Break	
1:55 – 2:35	Field Lab	Alternating between AI-Native Labs and Client Engagement Labs. AI labs require a documented rejection; client labs are role-played with instructors as stakeholders and are recorded for critique.
2:35 – 3:00	Sprint review	Demo, pull request review, evidence check and retrospective. Milestone sprints extend this segment.

Why the client conversation is protected live time

Data extraction can be learned alone. Telling a sponsor their data is worse than they believed, holding scope against a fifth requirement, or teaching an operations team to run a system they did not ask for — none of that can be practised from a video. It requires another person, in real time, who is allowed to be difficult. That is what the Client Engagement Labs are for, and it is the single largest reason this programme cannot be delivered asynchronously.

4. The Technology Stack

Tools are introduced when a project needs them, never as a standalone module. Each carries a depth code indicating expected independence. The stack below is what separates this programme from a full-stack curriculum.

Application layer

Tool	Depth	How it is used in the programme
TypeScript, React, Vite	M	The console and internal tooling layer. Strict typing from Sprint 04; components, forms, routing and state through Sprints 05–06.
Node.js 22, Express	M	Application services: typed APIs, layered architecture, structured logging and OpenAPI contracts in Sprint 07.
Tailwind, MUI, MUI X	H	Interface system and data grids for internal tools, with per-client theming from tokens.
TanStack Query, Redux Toolkit	H	Server cache and global client state, chosen by category rather than by habit.
Vitest, Playwright, MSW, Supertest	M	Component, integration, end-to-end and API testing across every sprint.

Data layer — the integration core

Tool	Depth	How it is used in the programme
PostgreSQL	M	Primary store. Modelling, migrations, indexes, EXPLAIN ANALYZE, window functions, transactions and row-level security across Sprints 08 and 10.
Prisma or Drizzle	M	Typed schema, migrations and queries against PostgreSQL.
Python 3.12, pandas, SQLAlchemy	M	The integration language, introduced in Sprint 12. Extraction, profiling, cleaning, bulk load and upsert with pytest coverage.
Airbyte	M	Managed connectors in Sprint 13, taught alongside a hand-built connector so learners understand what the tool abstracts.
SFTP, REST, SOAP, ODBC/JDBC	H	Legacy source access: Oracle and SQL Server drivers, XML and WSDL, fixed-width and delimited files, encoding failures.
Temporal	M	Durable orchestration in Sprint 14: workflows, activities, retries, timeouts, backfill, replay and compensating actions.
Great Expectations	M	Expectation suites and validation gates that block a bad load rather than reporting it afterwards.
DuckDB	H	Local analysis of extracts before they reach a warehouse.
MongoDB, Mongoose	H	Used where a document model earns its place, and justified by access pattern rather than preference.
Redis, BullMQ	M	Caching, idempotency keys, sessions, queued work with retries and dead-letter handling.

Deployment and infrastructure layer

Tool	Depth	How it is used in the programme
Docker, Compose	M	Multi-stage builds, non-root users, health checks and a reproducible local stack.
Kubernetes	M	Deployments, services, ingress, config maps, secrets, probes, resource limits and rolling updates in Sprint 16.
Helm	M	Charts with per-client values files, producing different deployments from one codebase.
Terraform	M	Providers, resources, state, modules and remote backends. A client-shaped environment provisioned and destroyed to prove reproducibility.
AWS	M	VPC, subnets, security groups, IAM, RDS, S3 and managed Kubernetes, provisioned end to end.
Microsoft Azure	H	VNet, Entra ID, Azure Database, Blob Storage and AKS — the equivalents that matter for Indian enterprise clients.
HashiCorp Vault	M	Dynamic credentials, secret rotation and runtime injection, with nothing committed to a repository.
GitHub Actions	M	Lint, type check, test, build, scan, plan, gated apply, smoke test and rollback.
Air-gapped delivery	H	Offline bundles, image tarballs, private registries and installation runbooks for environments with no egress.

Identity, tenancy and compliance

Tool	Depth	How it is used in the programme
Keycloak	H	The teaching identity provider: realms, clients, mappers and federation, integrated end to end in Sprint 09.
OIDC and SAML 2.0	M	Authorization code flow with PKCE, ID tokens and claims; SAML assertions, metadata exchange and IdP-initiated flows.
SCIM	H	User and group provisioning, including just-in-time provisioning from the client directory.
Okta, Microsoft Entra ID	E	Integration patterns for the two identity providers learners will most often meet in the field.
PostgreSQL row-level security	M	Tenant isolation enforced at the database boundary, with an automated cross-tenant leakage test suite.
Unleash	H	Feature flags, rollout strategies, kill switches and flag cleanup discipline.
SOC 2, ISO 27001, DPDP Act	H	Control mapping, audit logging, retention policy and evidence production — not certification.

AI layer

Tool	Depth	How it is used in the programme
Claude, Claude Code, Cursor	M	Primary assistants from Sprint 01: explanation, review, agentic refactors and repository-aware work under a disclosure discipline.
Anthropic API, Vercel AI SDK, Zod	M	Typed clients, structured output validated at runtime, streaming interfaces and model-failure test suites.
Model Context Protocol (MCP)	H	Tool definition with typed schemas, permissions, validation and audit logs; agent authorisation boundaries.
pgvector	M	Retrieval inside the primary database — chunking, embeddings, metadata filters, hybrid search and source attribution.
Azure OpenAI, AWS Bedrock	M	Private inference inside a client's cloud boundary, with data-handling terms understood before selection.
vLLM, Ollama	H	Self-hosted inference, quantization and hardware sizing where data cannot leave the premises.
Unstructured.io, Docling, OCR	M	Ingestion of real enterprise documents, including scanned records.
Presidio	M	PII detection and redaction at ingestion, verified rather than assumed.
Ragas, Promptfoo, Langfuse	M	Faithfulness and relevance scoring on a client-signed golden set, regression evaluation, tracing, latency and cost observability.

Observability, delivery and client-facing tooling

Tool	Depth	How it is used in the programme
OpenTelemetry	M	Traces, metrics, logs and context propagation across console, API, worker and pipeline.
Prometheus, Grafana, Loki	M	Instrumentation, dashboards, log aggregation and alert rules tied to defined SLOs.
Sentry	M	Error tracking, release health and regression detection.
PostHog	H	Product analytics, including self-hosted deployment for on-premises clients.
Metabase	H	Client-facing reporting and reconciliation dashboards a stakeholder can read without help.
Miro, Notion, Linear, Mermaid	M	Discovery workshops, briefs, statements of work, backlogs and architecture diagrams.
Loom	H	Asynchronous client communication: explaining a data-quality failure or a release in five minutes.

The AI Usage Charter

- Generated code is untrusted until reviewed, tested, understood and disclosed.
- Every learner maintains an AI Disclosure Log from Sprint 01 to graduation, recording what was generated, what was rejected, and how each accepted output was verified.

- Every AI-Native Lab requires at least one documented rejection with written reasoning.
- Anything a learner cannot explain in a sprint review is removed from the codebase before submission.
- Agent and tool-calling workflows use least privilege, explicit schemas, server-side authorization, bounded retries, audit logs and human approval for consequential actions.
- On client engagements, AI-assisted work is disclosed to the client. Schema inference and data profiling produced by a model are treated as hypotheses requiring verification against the source.

5. Depth of Coverage

Every topic in this document carries a depth code. The code describes the independence a learner is expected to reach, not the topic's importance in industry. It is the primary mechanism preventing the programme from over-claiming — which matters more here than usual, because the stack is broad by necessity.

Code	Definition
M — Mastery	The learner applies the skill independently and is formally assessed on it.
H — Hands-on	The learner implements it, but it is not treated as a major exit competency.
E — Exposure	Guided lab or demonstration. The learner should recognise it and use it with support.
A — Awareness	The learner knows what it is and when it might be used, without implementing it.

6. Programme Outcomes

Engineering outcomes

- Build accessible, responsive internal tooling using semantic HTML, modern CSS, React and TypeScript.
- Design typed REST services using Node.js and Express with layered architecture, structured logging and a verified OpenAPI contract.
- Model, index, migrate and query production PostgreSQL, and justify when a document store is warranted.
- Implement caching, idempotent writes, queued background work with dead-letter handling, and authenticated real-time features.
- Write unit, component, integration, contract and end-to-end tests throughout development.

Data integration outcomes

- Extract, profile, clean and load data in Python with tests and idempotent re-run behaviour.
- Build custom connectors with pagination, authentication, incremental cursors, persisted state and drift detection.
- Ingest from database, API and file-drop sources into a single normalised model with reconciliation reporting.
- Orchestrate durable pipelines with retries, timeouts, backfill, replay and compensating actions.
- Implement data-quality gates that block a bad load, with a quarantine path and operator remediation.
- Produce a source-system brief for an undocumented database, written for a non-technical stakeholder.

Deployment and operations outcomes

- Package a system so it runs identically on a laptop, a client cluster and an air-gapped server.
- Deploy to Kubernetes with Helm charts producing per-client variations from one codebase.
- Provision reproducible client infrastructure as Terraform code, with remote state and a destroy-and-rebuild proof.
- Federate an application to an enterprise identity provider over OIDC or SAML, with SCIM-ready provisioning.
- Isolate tenants at the database boundary and prove it with an automated leakage test suite.
- Instrument a distributed system, diagnose an incident from telemetry alone, and write a blameless postmortem.
- Produce audit logging, retention policy and control mapping as compliance evidence.

AI engineering outcomes

- Deploy private inference inside a client cloud boundary, or self-hosted where data cannot leave the premises.
- Build permission-aware retrieval where access is enforced per user, per tenant and per document at query time.
- Ingest real enterprise documents including scanned records, with PII detection and redaction verified at ingestion.
- Evaluate AI features against a client-signed golden set with faithfulness, relevance and a failure-mode catalogue.
- Produce cost, latency and token budgets, and a business case with honest limitations.
- Use AI for development responsibly, with review, verification, testing, disclosure and documented rejection.

Client engagement outcomes

- Run a discovery conversation that reaches the problem behind the stated request.
- Produce a discovery brief, solution architecture, estimate with assumptions and exclusions, and a scoped statement of work.
- Hold scope against a difficult stakeholder and frame trade-offs without becoming either rigid or agreeable.
- Deliver bad news — a data-quality failure, a missed assumption, a limitation — clearly and without assigning blame.
- Prove value inside a week with a deliberately disposable prototype, and state plainly what was faked.
- Hand over completely: runbook, administrator guide, enablement session, support model, and proof that someone else can operate the system unaided.

7. The 24-Week Curriculum Map

Forty learning blocks are delivered across twenty-four calendar weeks as twenty sprints. Sixteen sprints run in a single calendar week; four deeper sprints run across two. Phase I is deliberately compressed relative to a general full-stack programme, because forward deployed engineers build internal tooling rather than consumer interfaces — the freed weeks are reallocated to integration, deployment and client work.

Most projects extend the same sprint repository rather than starting a new application. Seven are designated major portfolio milestones and receive a deeper review.

PHASE I · Calendar Weeks 1–8 · Frontend Foundations for Internal Tooling

Sprint	Weeks	Focus	Implementation project outputs
01	1	Web foundations, semantic markup, Git	Semantic admin console shell; responsive console foundation
02	2	Responsive UI, design systems, theming	Operations dashboard; themeable console UI kit
03	3–4	JavaScript foundations, async, browser apps	Logic toolkit; modular API explorer · Technical track opens
04	5	TypeScript and typed API integration	Typed domain toolkit; typed API application · Milestone 1
05	6–7	React: components, forms, routing	Component gallery; routed internal application
06	8	State, testing, accessibility, performance	State refactor; tested accessible console · Milestone 2

PHASE II · Calendar Weeks 9–15 · Backend, Data Modelling and Enterprise Identity

Sprint	Weeks	Focus	Implementation project outputs
07	9–10	Node, Express and API architecture	Typed REST API; layered and documented API
08	11–12	PostgreSQL, relational modelling, polyglot	Relational domain model; persistent data API · Milestone 3
09	13	Auth, authorization, enterprise identity	Account lifecycle; enterprise SSO integration · Placement opens
10	14	Multi-tenancy, configurability, feature control	Tenant-isolated API; configurable multi-client console
11	15	Caching, jobs, real-time, production workflows	Caching and idempotency; jobs and real-time · Milestone 4

PHASE III · Calendar Weeks 16–19 · Data Integration and Client Discovery

Sprint	Weeks	Focus	Implementation project outputs
12	16	Python for data engineering	Messy spreadsheet extractor; API-to-database loader
13	17	Connectors and ingestion from legacy systems	Custom source connector; multi-source ingestion layer
14	18	Orchestration, data quality, reconciliation	Durable ingestion workflow; data quality and reconciliation gate
15	19	Discovery, scoping, 72-hour prototype	Discovery brief and SOW; prototype and go/no-go · Milestone 5

PHASE IV · Calendar Weeks 20–24 · Enterprise Deployment, AI and Handover

Sprint	Weeks	Focus	Implementation project outputs
16	20	Containers, Kubernetes, reproducible environments	Containerised stack; Helm multi-client release
17	21	Infrastructure as code, enterprise cloud, secrets	Terraform client environment; gated delivery pipeline
18	22	AI on client data: private inference, RAG, evaluation	Permission-aware RAG; AI evaluation and business case · Milestone 6
19	23	Observability, reliability, compliance evidence	Instrumented system; incident response and compliance pack
20	24	Handover, enablement, client capstone defence	Handover pack and enablement; client capstone · Milestone 7

8. The Field Skills Track

Client engagement capability is built through Client Engagement Labs, which occupy the Field Lab slot in the Sunday studio and alternate with AI-Native Labs across the programme. Each is role-played with instructors or alumni acting as stakeholders, recorded, and critiqued as a group. Nine of the twenty sprints carry one.

Sprint	Client Engagement Lab	The skill being built
02	Client branding drill	Configurability as a design constraint — a rebrand delivered through tokens alone, with no component edits.
10	Configuration versus customisation	Scope discipline. Three client requirement sets triaged into configuration, feature flag or genuine product change, with a written decision memo.
13	The undocumented source	Investigation and written communication. A source-system brief produced from database access and no documentation, written for a non-technical stakeholder.
14	Delivering bad news	Explaining a material data-quality failure to a sponsor: what is wrong, what it means, what is recommended, what is needed — without assigning blame.
15	Scope defence	Holding a statement of work before a panel playing procurement, IT security and the business sponsor, with conflicting priorities.
16	The constrained environment	Producing a deployment plan within a client's real infrastructure limits, and knowing which constraints to push back on.
17	The security questionnaire	Completing an enterprise security review honestly, marking the items not yet satisfied with a remediation plan and timeline.
19	Operational handover rehearsal	Teaching a client operations team to run the system, then being graded on whether they succeed unaided — not on the quality of the explanation.
20	The quarterly business review	Presenting outcomes, value, failures, roadmap and cost to a client sponsor. Graded on honesty about failure as much as on the outcome.

Why these are assessed live and recorded

A written answer to "how would you tell a client their data is unusable" bears no relationship to doing it while a stakeholder becomes defensive in real time. The labs are recorded so learners can watch themselves — which is uncomfortable and is the point. Assessment uses the Discovery, Scoping and Client Communication rubric, and the recordings form part of the portfolio evidence.

9. The Technical Assessment Track

This track occupies the first forty-five minutes of every Sunday studio and generates five problems per week of self-paced work. It deliberately diverges from a standard interview-preparation curriculum after Calendar Week 12.

Period	Format	Content
Weeks 3–5	DSA Level 1	Big-O basics; arrays, strings, objects and hash maps; frequency counting [M]. Sets,

Period	Format	Content
		linear search and basic sorting [H].
Weeks 6–8	DSA Level 2	Two pointers, sliding window and binary search [M]. Prefix sums, stack, queue and linked list [H]. Checkpoint 1 at the end of Week 8.
Weeks 9–12	DSA Level 3	Recursion, trees, BFS and DFS, BST, heap and priority queue [H]. Checkpoint 2 at the end of Week 12.
Weeks 13–18	System design	Identity and access architecture, multi-tenancy, caching and back-pressure, integration coupling, durable execution, deployment topology and blast radius.
Weeks 19–22	Case interview	Ambiguous business problems, structured framing, hypothesis-driven questioning, and scoping an AI project against an unrealistic internal expectation.
Weeks 23–24	Panel practice	Full system-design and case panels with written feedback. Checkpoint 3 assessed at the end of Week 24.

Why DSA stops at Level 3

Forward deployed interview loops are weighted toward system design, integration architecture and case discussion. Dynamic programming and advanced graph algorithms appear rarely, and the twelve weeks they would consume are worth more spent on the skills the loop actually tests. Learners intending to interview at organisations with algorithm-heavy screens should treat Levels 4 and 5 as independent preparation; the material remains available on the platform and instructors will review submissions on request.

10. Sprint-by-Sprint Specification

The full specification for all twenty sprints follows. Each entry states what the learner covers alone during the week, what happens in each of the two live studios, the projects produced, the Field Lab, the technical track position, the required evidence, the exit condition and the scope guardrail.

PHASE I · Frontend Foundations for Internal Tooling

SPRINT 01 · **Calendar Week 1**

Web Foundations, Semantic Markup and Git Discipline

Establish how the web works, build accessible semantic pages, and set up the repository habits that every client engagement will depend on.

Self-paced Mon–Fri	– Request–response lifecycle, DNS, hosting, HTTP methods and status codes [H] – Semantic document landmarks, headings, tables and forms [M] – Cascade, specificity, box model and typography scales [M] – Flexbox, Grid and mobile-first media queries [M] – Git branches, commits, pull requests and README authoring [M] – DevTools, command line and local environment setup [H] – AI and GenAI fundamentals: LLM basics, prompting, limitations, verification [H]
Saturday Implementation Studio	– Readiness check on the week's self-paced syntax; blocker clinic – Live build: semantic admin console shell from a blank repository – DevTools walkthrough — DOM, network panel, accessibility tree – Learner build sprint with keyboard-navigation audit
Sunday Field Engineering	– Engineering communication primer: writing a status update a client executive will actually read – Live build: extend the console into a responsive multi-section layout – Git collaboration live — branch, pull request, review response, merge – Deploy to static hosting in-session; verify the live URL – Field Lab and sprint review
Projects	– Implementation Project 1 — Semantic Admin Console Shell. A single-page internal console with meaningful document structure, accessible navigation, a data table and a validated form. – Implementation Project 2 — Responsive Console Foundation. Extend Project 1 into a responsive multi-section console published through static hosting.
Field Lab (AI-Native)	– Toolchain: Claude · Claude Code · Cursor – Prompt-and-critique drill. Learners ask Claude to explain a semantic markup decision, verify it against MDN, and record where the model was right, vague or wrong. First entry in the AI Disclosure Log.
Technical track	Technical track not yet open. Pre-track primer: reading Big-O notation and recognising cost language in vendor documentation.
Evidence	Repository, semantic audit notes, DevTools screenshot, responsive evidence, keyboard review, deployment link, one pull request with a review response, AI disclosure entry.
Sprint exit	Learner can explain the browser request lifecycle and deliver a semantic responsive page through a clean Git workflow.
Scope guardrail	HTML, CSS, browser tools and Git only. No frameworks. No AI-generated markup accepted without line-by-line explanation.

SPRINT 02 · Calendar Week 2

Responsive Interface Engineering and Design Systems

Build consistent internal interfaces from reusable layout rules and visual tokens, at the depth an internal tool needs rather than a marketing site.

Self-paced Mon–Fri	– Intrinsic sizing, min/max/clamp and container queries [H] – Advanced Grid, overflow, navigation, table, form, empty and loading layouts [M] – Design tokens, colour roles and CSS custom properties [M] – Tailwind configuration and utility composition [M] – MUI components, theming and MUI X data grid [H] – Image and font performance fundamentals [E]
Saturday Implementation Studio	– Readiness check; live build of an operations dashboard shell across breakpoints – Content-stress testing in-session — short, long, empty and error states – Learner build sprint producing a viewport test matrix
Sunday Field Engineering	– Technical track: complexity intuition and cost language – Live build: design tokens to Tailwind theme, then the same system expressed through MUI – Accessibility review — contrast, focus order, keyboard traps — fixed live – Field Lab and sprint review
Projects	– Implementation Project 3 — Responsive Operations Dashboard. A dashboard shell that adapts across breakpoints and stays stable with short, long, empty and error content. – Implementation Project 4 — Themeable Console UI Kit. A small reusable interface system with tokens, applied to the console and prepared for per-client rebranding.
Field Lab (Client Engagement)	– Toolchain: Figma · Miro · Claude – Client branding drill. Learners are handed a fictional client's brand sheet mid-session and must retheme the console through tokens alone, with no component edits. Introduces configurability as a design constraint rather than a later refactor.
Technical track	Technical track not yet open.
Evidence	Viewport test matrix, documented tokens, reusable components, accessibility review, retheme demonstration, live deployment.
Sprint exit	Learner can implement a consistent, rethemeable interface system rather than isolated pages.
Scope guardrail	Limit the system to components the console actually needs. No general-purpose component library.

SPRINT 03 · Calendar Weeks 3–4

JavaScript Foundations, Async and Browser Applications

Develop reliable programming habits with functions, collections, modules and asynchronous execution, then handle real network and UI failure states.

Self-paced Mon–Fri	– let/const, primitives, operators, control flow, functions and scope [M] – Closures, pure functions and side effects [M] – map / filter / find / reduce, object updates and destructuring [M] – Map, Set, local storage and the DOM API [M] – Breakpoints, watch expressions and debugging method [M] – ES modules, npm scripts and project structure [M] – Event loop, Promises, async/await, sequencing and try/catch [M] – Parallel work, timeouts, retries and cancellation [H] – Loading, empty, error and retry UI states [M] – Stacks, queues, frequency maps and Big-O intuition [H]
Saturday Implementation Studio	– Live build: JavaScript logic toolkit — validation, pricing, filtering, rule engines – Debugging clinic — instructor introduces defects live, learners locate them under time pressure – Week 2: live refactor into modules with feature boundaries and package scripts
Sunday Field Engineering	– Technical Clinic opens — Level 1: complexity analysis, arrays, strings, hash maps – Live build: interactive DOM task board with filters and persistence – Week 2: public API explorer with cancellation, pagination and normalised errors; network debugging under throttling and 500s – Field Lab and sprint review
Projects	– Implementation Project 5 — JavaScript Logic Toolkit. Small utilities for validation, pricing, filtering and rule-based decisions with clear inputs, outputs and test cases. – Implementation Project 6 — Modular API Explorer. A modular browser application consuming a public API with cancellation, persistence, and recoverable loading, empty and error states.
Field Lab (AI-Native)	– Toolchain: Claude (Socratic mode) · Cursor · GitHub Copilot – AI-assisted generation, explanation and debugging. Learners configure Claude as a hint-ladder tutor rather than an answer engine, and compare Copilot completion against a hand-written implementation on correctness and readability.
Technical track	Technical Level 1 opens [M]: Big-O basics, arrays, strings, objects and hash maps, frequency counting. Sets, linear search and basic sorting [H]. Five async problems per week with written complexity justification.
Evidence	Modular functions, test cases, debugging record, module map, working API application, network failure notes, pull request, five problem solutions with complexity notes.
Sprint exit	Learner can translate requirements into small functions and build a modular asynchronous browser application without a framework.
Scope guardrail	Practical patterns only. Recursion and advanced algorithms remain awareness topics at this stage.

SPRINT 04 · Calendar Week 5

TypeScript and Typed API Integration

Add strict static typing and runtime validation at every application boundary — the discipline that makes integration code survivable.

Major Milestone 1 falls in this sprint.

Self-paced Mon–Fri	– Primitive and object types, functions, unions, literals and narrowing [M] – Interfaces, type aliases, generics and utility types [M] – Strict compiler configuration; avoiding unsafe any [M] – Typing API responses and forms; schema validation at the boundary [M] – Typed error normalisation, auth headers, retry and pagination patterns [M]
Saturday Implementation Studio	– Live migration: the JavaScript logic toolkit converted to strict TypeScript, error by error – Domain modelling in-session — data, errors and configuration as types – Reading compiler output as a design signal
Sunday Field Engineering	– Technical Clinic — Level 2 opens: two pointers and sliding window – Live build: the API explorer rebuilt in strict TypeScript with Zod runtime schemas – Compile-time versus runtime — a deliberate failure demonstration – Field Lab, milestone clinic and sprint review
Projects	– Implementation Project 7 — Typed Domain Toolkit. The logic toolkit converted to strict TypeScript with domain data, errors and configuration modelled as types. – Implementation Project 8 — Typed API Application — Major Milestone 1. The API explorer rebuilt as a strict TypeScript application with runtime validation and resilient network handling.
Field Lab (AI-Native)	– Toolchain: Claude · Cursor · Anthropic API · Zod · Vercel AI SDK – Typed GenAI client. Learners build an LLM client returning structured output validated by a Zod schema, then write model-failure test cases: malformed JSON, hallucinated fields, truncation, refusals. Prompt and model version are treated as configuration.
Technical track	Technical Level 2 opens [M]: two pointers, sliding window, binary search. Prefix sums, stack, queue and linked list [H].
Evidence	Strict compiler result, typed modules, documented type decisions, typed API client, runtime schemas, tests, deployment, structured-output failure suite.
Sprint exit	Learner can distinguish compile-time types from runtime validation and build a reliable typed API-driven application.
Scope guardrail	Types clarify domain behaviour. Avoid advanced type-level programming for its own sake.

SPRINT 05 · Calendar Weeks 6–7

React with TypeScript: Components, Forms and Routing

Build typed component-based internal applications with predictable data flow, validated workflows and URL-driven navigation.

Self-paced Mon–Fri	– Vite setup, environment variables and project structure [H] – React rendering, JSX, composition, typed props and events [M] – useState, derived values, immutable updates and lifting state [M] – Lists and keys, conditional rendering, effects for external synchronisation [M] – Controlled forms, React Hook Form and schema validation [M] – Accessible errors, focus management and multi-step flows [M] – Routes, parameters, nested layouts, not-found and protected-route UX [M] – URL search parameters, lazy routes and error boundaries [H] – TanStack Query: server cache, invalidation and mutations [H]
Saturday Implementation Studio	– Live build: typed component gallery with documented props, variants and states – Component API design — what belongs in props and what does not – Week 2: multi-step onboarding form with draft persistence and accessible error recovery
Sunday Field Engineering	– Technical Clinic — Level 2: binary search and linked-list patterns – Live build: React record browser with search, filters, selection and error handling – Week 2: routed application with URL-driven filters and mocked authentication states – Field Lab and sprint review
Projects	– Implementation Project 9 — Typed Component Gallery. Accessible reusable components with documented props, variants, states and composition examples. – Implementation Project 10 — Routed Internal Application. A routed React application combining a record browser and a validated multi-step workflow, with URL-driven filters and a typed service layer.
Field Lab (AI-Native)	– Toolchain: Claude · Cursor Composer · Vercel AI SDK · Playwright – AI-assisted component generation under review, then AI interaction UX. Learners generate with Composer, run a structured review pass (prop types, accessibility, re-render behaviour, dead code), then build a streaming assistant surface with fallback UX for invalid, unsafe and truncated output.
Technical track	Technical Level 2 continues [M / H]: binary search variants, prefix sums, stack, queue, linked list.
Evidence	Typed component API, gallery, keyboard checks, route map, URL-state tests, form submission-state tests, component tests, demo.
Sprint exit	Learner can build a routed typed React application with validated workflows and deliberate state ownership.
Scope guardrail	Authentication remains mocked. Redux is deferred until a demonstrated need appears in Sprint 06.

SPRINT 06 · Calendar Week 8

State Architecture, Testing, Accessibility and Performance

Select state tools by category, verify user-visible behaviour, and improve measured quality — closing the frontend phase with evidence rather than features.

Major Milestone 2 falls in this sprint. Frontend completion gate.

Self-paced Mon–Fri	– Local, derived, form, URL, global client and server state taxonomy [M] – Context boundaries; Redux Toolkit and RTK Query [H] – Caching, invalidation and mutation patterns [H] – React Testing Library, user-event, router and state tests [M] – API mocking with MSW; deterministic fixtures [M] – Render profiling, lazy loading, bundle review, axe and Lighthouse [H]
Saturday Implementation Studio	– Live refactor: every state category assigned a deliberate owner – Server-cache layer introduced with invalidation demonstrated live – Architecture decision note written in-session
Sunday Field Engineering	– Technical Clinic — Level 3 opens: recursion and tree traversal – Live build: high-value component and integration tests written against real user behaviour – Profiling session — measure, optimise, measure again – Field Lab, milestone review and sprint review
Projects	– Implementation Project 11 — State Architecture Refactor. The routed application refactored so each state category has a deliberate owner and server data uses an appropriate cache layer. – Implementation Project 12 — Tested Accessible Console — Major Milestone 2. A deployed, tested and accessible internal console with a documented architecture decision record and one measured performance improvement.
Field Lab (AI-Native)	– Toolchain: Claude Code · Vitest · Playwright · Promptfoo – AI-generated tests, human-verified. Claude Code proposes a suite; learners delete every test that passes for the wrong reason, then add the cases the model missed. Introduction to golden test sets with Promptfoo.
Technical track	Technical Level 3 opens [H]: recursion, trees, BFS and DFS. Checkpoint 1 assessed at the end of this week, covering Levels 1 and 2.
Evidence	State inventory, architecture decision record, cache behaviour test, test suite, MSW mocks, accessibility report, profile evidence, deployment, deleted-test justification log, Checkpoint 1 result.
Sprint exit	Learner can justify state choices, verify behaviour, and ship an accessible tested frontend. Frontend phase complete.
Scope guardrail	Optimisation must follow measurement. Real-time integration deferred until the backend exists.

PHASE II · Backend, Data Modelling and Enterprise Identity

SPRINT 07 · Calendar Weeks 9–10

Node, Express and API Architecture

Build a typed REST service, then refactor it into maintainable layers with a published contract and tests at the right levels.

Self-paced Mon–Fri	– Event-driven architecture, non-blocking I/O, process lifecycle and signals [M] – Node 22 LTS, scripts, dependency hygiene, environment validation and graceful shutdown [M] – Request lifecycle, middleware order, routes and controllers [M] – Status codes, validation, standardised errors, pagination and filtering [M] – Controllers, services, repositories, domain errors and dependency boundaries [M] – Structured logging with Pino, request IDs and correlation [M] – OpenAPI documentation and contract verification [M] – Supertest integration tests, fixtures and deterministic data [M]
Saturday Implementation Studio	– Live build: typed Node service with validated configuration, structured startup and shutdown – Debugging a Node process live — source maps, signals, leaked handles – Week 2: live refactor extracting framework-coupled logic into testable services
Sunday Field Engineering	– Technical Clinic — Level 3: BST, heap and priority queue – Live build: typed Express REST API with validated requests and centralised errors – Week 2: OpenAPI contract published and verified against the running API; structured logging and request IDs added – Field Lab and sprint review
Projects	– Implementation Project 13 — Typed Express REST API. A REST API with validated requests, consistent responses, pagination, centralised errors and route tests. – Implementation Project 14 — Layered and Documented API. The API refactored into controllers, services and repositories, with structured logging, request IDs, a verified OpenAPI contract and integration tests.
Field Lab (AI-Native)	– Toolchain: Claude Code · MCP · Postman · CodeRabbit – Server-side AI gateway and first MCP exposure. Learners build a provider-abstracted AI gateway route, then define one internal tool with a typed schema and permission boundary. Tests must cover invalid tool arguments and denied actions.
Technical track	Technical Level 3 continues [H]: trees, BFS/DFS, heap and priority queue.
Evidence	Environment schema, health check, shutdown demonstration, API collection, route tests, architecture diagram, OpenAPI file, structured logs, integration tests, MCP tool schema with denied-action tests.
Sprint exit	Learner can build, layer, document and test an Express API and explain the Node process operating it.
Scope guardrail	In-memory data only. Databases, authentication and queues are deferred to the following sprints.

SPRINT 08 · Calendar Weeks 11–12

PostgreSQL, Relational Modelling and Polyglot Persistence

Make PostgreSQL the primary store — the reality of enterprise deployment — then place MongoDB deliberately where a document model earns its keep.

Major Milestone 3 falls in this sprint.

Self-paced Mon–Fri	– Tables, keys, relationships, normalisation, joins and constraints [M] – PostgreSQL types, transactions, isolation levels and connection pooling [M] – Prisma or Drizzle: schema, migrations, typed queries and seeding [M] – Indexes, composite indexes, EXPLAIN ANALYZE and query plans [M] – Window functions, CTEs and set operations [H] – MongoDB documents, embedding versus referencing, access-pattern modelling [H] – Mongoose schemas, validation, aggregation pipelines and isolated database tests [H] – Choosing a store: pgvector versus a dedicated vector database [H]
Saturday Implementation Studio	– Live modelling: the same client domain expressed relationally and as documents, side by side – SQL lab in-session — joins, CTEs and query plans on a real engine – Week 2: migration authoring, rollback and seed data built live
Sunday Field Engineering	– Technical Clinic — Level 3 consolidation; Checkpoint 2 preparation – Live build: in-memory storage replaced with PostgreSQL, indexes verified through EXPLAIN – Week 2: one aggregation reporting endpoint benchmarked live; polyglot trade-off debate – Field Lab, milestone clinic and sprint review
Projects	– Implementation Project 15 — Relational Domain Model and SQL Lab. A normalised PostgreSQL schema for a client domain, with migrations, seed data, joins, CTEs and documented query plans. – Implementation Project 16 — Persistent Data API — Major Milestone 3. The layered API backed by PostgreSQL with verified indexes, one reporting endpoint, transactional writes and integration tests, plus a written polyglot trade-off analysis.
Field Lab (AI-Native)	– Toolchain: pgvector · Anthropic / OpenAI embeddings · LlamaIndex.TS · Claude Code – Retrieval inside the primary database. Learners build ingestion and retrieval end to end using pgvector — chunking, embeddings, metadata filters, hybrid search, source attribution — then test retrieval quality rather than assuming it. Update and delete behaviour must be correct, not just insert.
Technical track	Technical Level 3 completes [H]. Checkpoint 2 assessed at the end of Calendar Week 12, covering Level 3.
Evidence	Relational schema, migrations, query plans, seed script, persistent API, transactional write test, reporting endpoint, retrieval-quality report with failure cases, trade-off analysis, Checkpoint 2 result.
Sprint exit	Learner can design, index, migrate and query a production PostgreSQL schema and justify when a document store is warranted.
Scope guardrail	PostgreSQL is the default. MongoDB must be justified by an access pattern, not by preference.

SPRINT 09 · Calendar Week 13

Authentication, Authorization and Enterprise Identity

Move beyond application-owned login to the identity reality of client environments: their directory, their protocol, their compliance requirements.

Placement and client-engagement activity begins this week.

Self-paced Mon–Fri	– Registration, hashing, sessions, access and refresh tokens, rotation and revocation [M] – Roles, permissions, ownership, default deny and object-level authorization tests [M] – OIDC authorization code flow with PKCE; ID tokens, claims and scopes [M] – SAML 2.0 assertions, metadata exchange and IdP-initiated flows [H] – Keycloak realms, clients, mappers and federation [H] – SCIM user and group provisioning; just-in-time provisioning [H] – Okta and Microsoft Entra ID integration patterns [E] – CORS, secure headers, rate limiting, secrets handling and dependency review [M]
Saturday Implementation Studio	– Live build: complete account lifecycle with audit events and failure-case tests – Live integration: the application wired to Keycloak over OIDC, tokens inspected in the browser – Threat-modelling session on the learner's own API
Sunday Field Engineering	– Technical Clinic — system design opens: identity and session architecture – Live build: SAML federation configured against a test IdP; claim-to-role mapping – Authorization matrix produced and then attacked in pairs – Field Lab and sprint review
Projects	– Implementation Project 17 — Secure Account Lifecycle. Registration, login, logout, recovery, session or token management, audit events and failure tests. – Implementation Project 18 — Enterprise SSO Integration. The API and console federated to Keycloak over OIDC with SAML fallback, claim-to-role mapping, SCIM-ready user model and a documented authorization matrix.
Field Lab (AI-Native)	– Toolchain: Claude · Snyk · MCP · Promptfoo – AI security under enterprise identity. Learners write a threat model covering direct and indirect prompt injection, data leakage, tool authorisation and per-user retrieval scoping, then build adversarial tests that must fail safely. Retrieval and tool calls are tested per identity, not per application.
Technical track	Technical track shifts to system design and case-interview practice [M]: identity, session and access-control architecture. Weekly problem sets continue at Level 3 difficulty.
Evidence	Threat notes, account-state tests, secure token settings, audit events, working OIDC login, SAML assertion capture, claim mapping, authorization matrix, adversarial AI test suite, security checklist.
Sprint exit	Learner can federate an application to an enterprise identity provider and prove authorization is enforced at the backend boundary.
Scope guardrail	One IdP integrated end to end beats three configured superficially. Keycloak is the teaching IdP; Okta and Entra remain pattern-level.

SPRINT 10 · Calendar Week 14

Multi-Tenancy, Configurability and Feature Control

Build once, deploy to many. Design for tenant isolation, per-client configuration and controlled rollout before the first client asks for something different.

Self-paced Mon–Fri	– Tenancy models: shared schema, schema-per-tenant, database-per-tenant [M] – PostgreSQL row-level security, policies and tenant-scoped connections [M] – Tenant context propagation through middleware, jobs and logs [M] – Configuration-driven behaviour: JSON Schema, validation and defaults [M] – Config-driven UI with react-jsonschema-form and theme tokens [H] – Feature flags with Unleash: rollout strategies, kill switches and cleanup [H] – Rules engines and workflow configuration versus code branching [H] – Per-tenant data export, deletion and residency obligations [H]
Saturday Implementation Studio	– Live build: row-level security applied to the persistent API; cross-tenant leakage tested and blocked – Tenant context threaded through requests, background jobs and log lines – Live drill — an intentionally leaky query, found and fixed as a group
Sunday Field Engineering	– Technical Clinic — system design: multi-tenant architecture trade-offs – Live build: per-tenant configuration and theming driven from a validated JSON Schema – Feature flags wired with a kill switch demonstrated live – Field Lab and sprint review
Projects	– Implementation Project 19 — Tenant-Isolated API. Row-level security, tenant-scoped queries, context propagation and an automated cross-tenant leakage test suite. – Implementation Project 20 — Configurable Multi-Client Console. Per-tenant configuration, theming and workflow variation driven by validated configuration, with feature flags and a working kill switch.
Field Lab (Client Engagement)	– Toolchain: Unleash · react-jsonschema-form · Miro · Notion – Configuration versus customisation. Learners receive three fictional client requirement sets and must decide, for each request, whether it is configuration, a feature flag, or a genuine product change — then write the decision memo they would send the account team. Scope discipline is the graded skill.
Technical track	System design: multi-tenant data isolation, noisy-neighbour and per-tenant cost. Weekly problem sets continue.
Evidence	RLS policies, leakage test suite, tenant context in logs, configuration schema, three-client theming demonstration, feature flag with kill switch, configuration-versus-customisation decision memo.
Sprint exit	Learner can isolate tenants at the database boundary and vary client behaviour through configuration rather than forks.
Scope guardrail	One tenancy model implemented properly. Database-per-tenant remains a documented comparison.

SPRINT 11 · Calendar Week 15

Production Workflows, Caching, Jobs and Real-Time

Implement the stateful infrastructure that separates a demo from a system a client can run on Monday morning.

Major Milestone 4 falls in this sprint.

Self-paced Mon–Fri	– Redis keys and TTLs, cache-aside, invalidation, sessions and idempotency keys [M] – BullMQ queues, workers, retries, backoff and dead-letter handling [M] – Validated file upload, object storage and streaming patterns [H] – Transactional email and external-service integration with provider abstraction [H] – Socket.IO events, rooms, authentication, reconnection and presence [H] – Scheduled work, locking and exactly-once concerns [H] – LangChain.js orchestration and Langfuse tracing for AI workflows [H]
Saturday Implementation Studio	– Live build: Redis-backed caching with invalidation, plus idempotent write endpoints – Live build: a background job with retries and a dead-letter queue – Failure drill — the external service is switched off mid-demo
Sunday Field Engineering	– Technical Clinic — system design: queues, caching and back-pressure – Live build: an authenticated real-time feature with reconnection handling – Metrics and logs inspected under load – Field Lab, milestone clinic and sprint review
Projects	– Implementation Project 21 — Caching and Idempotent Writes. Redis-backed caching with explicit invalidation rules, plus idempotency keys on write endpoints and a documented cache policy. – Implementation Project 22 — Background Jobs and Real-Time Feature — Major Milestone 4. A queued workflow with retries and dead-letter handling, plus one authenticated real-time feature, both with failure tests and an operational note.
Field Lab (AI-Native)	– Toolchain: BullMQ + Redis · MCP · Langfuse · Claude Code – Agentic workflow orchestration. Learners run an agent as a background job with queue, retry, dead-letter and audit handling, then attempt to make it misbehave. Tool authorisation, stopping rules and human approval for consequential actions are implemented, not described.
Technical track	System design: caching strategy, queue depth, back-pressure and failure isolation.
Evidence	Cache policy note, invalidation tests, idempotency tests, queued workflow, dead-letter evidence, real-time feature, reconnection test, agent audit log with a documented failed attack.
Sprint exit	Learner can deliver caching, queued work and real-time features with explicit failure handling and operational visibility.
Scope guardrail	Uploads and email are optional pathways. Redis and queues are required; everything else is choose-one.

PHASE III · Data Integration and Client Discovery

SPRINT 12 · **Calendar Week 16**

Python for Data Engineering

Add the second language the integration layer runs on. Node builds the product; Python moves the client's data.

Self-paced Mon–Fri	– Python 3.12 syntax, typing, virtual environments and uv [M] – Project structure, packaging, logging and CLI entry points [M] – pandas: loading, reshaping, joining, cleaning and profiling tabular data [M] – openpyxl and csv: real-world spreadsheet extraction and its failure modes [M] – requests and httpx: paginated, authenticated and rate-limited API consumption [M] – SQLAlchemy: connections, reflection, bulk load and upsert [H] – pytest, fixtures and testing data transformations [M] – DuckDB for local analysis of extracts before they reach a warehouse [H]
Saturday Implementation Studio	– Live build: a Python extraction script against a deliberately messy spreadsheet — merged cells, inconsistent dates, trailing whitespace, mixed types – Data profiling in-session: what is actually in this file versus what the client said is in it – Learner build sprint with a written data-quality observation list
Sunday Field Engineering	– Technical Clinic — system design: batch versus streaming, and where transformation belongs – Live build: a paginated authenticated API extract loaded into PostgreSQL through SQLAlchemy – Upsert and re-run behaviour proven live — the same script run twice must not duplicate – Field Lab and sprint review
Projects	– Implementation Project 23 — Messy Spreadsheet Extractor. A tested Python extractor that ingests a deliberately malformed spreadsheet, profiles it, normalises it, and reports every row it rejected and why. – Implementation Project 24 — API-to-Database Loader. A paginated, authenticated, rate-limit-aware API extract loaded idempotently into PostgreSQL with pytest coverage and a re-run safety proof.
Field Lab (AI-Native)	– Toolchain: Claude Code · pandas · DuckDB · Cursor – AI-assisted data profiling. Learners point Claude Code at an unfamiliar dataset to generate a profiling report and candidate schema, then verify every inference against the raw file — documenting each place the model guessed wrong. Schema inference is treated as a hypothesis, not an answer.
Technical track	System design: extraction architecture, incremental load and idempotency. Weekly problem sets continue.
Evidence	Python project with packaging and logging, extractor with rejection report, data-quality observations, API loader, upsert proof, pytest suite, AI inference verification log.
Sprint exit	Learner can extract, profile, clean and load client data in Python with tests and idempotent re-run behaviour.
Scope guardrail	Python at hands-on depth for the data layer. Node remains the application language; this is not a second full stack.

SPRINT 13 · Calendar Week 17

Connectors and Ingestion from Systems You Do Not Own

The defining daily work of the role: getting data out of legacy systems with no documentation, no support and no ability to change them.

Self-paced Mon–Fri	– Airbyte: sources, destinations, connection configuration and sync modes [M] – Building a custom connector: pagination, auth, incremental cursors and state [M] – SFTP and file-drop ingestion: watching, locking, archiving and partial files [M] – REST and SOAP integration; XML parsing and WSDL basics [H] – ODBC and JDBC access to Oracle and SQL Server; driver and network constraints [H] – Fixed-width, delimited and EDI-style formats; encoding and line-ending failures [H] – Schema inference, drift detection and contract negotiation with the source owner [M] – Change data capture concepts and watermark-based incremental sync [H]
Saturday Implementation Studio	– Live build: an Airbyte connection from a legacy SQL source into PostgreSQL, then the same job written by hand to expose what the tool was doing – SFTP ingestion with partial-file and duplicate-delivery handling – Live failure drill — the source schema changes without warning mid-sync
Sunday Field Engineering	– Technical Clinic — system design: integration architecture and coupling to source systems – Live build: incremental sync with watermarks, replay and backfill – Reconciliation in-session: proving row counts and totals match the source – Field Lab and sprint review
Projects	– Implementation Project 25 — Custom Source Connector. A hand-built connector with pagination, authentication, incremental cursors, persisted state and drift detection, tested against a source that changes underneath it. – Implementation Project 26 — Multi-Source Ingestion Layer. Three sources — database, API and file drop — ingested into a single normalised model with reconciliation reporting and documented source assumptions.
Field Lab (Client Engagement)	– Toolchain: Airbyte · Claude · Notion · Mermaid – The undocumented source. Learners are given database access and no documentation, and must produce a source-system brief: what the tables mean, which fields are trustworthy, what the client believes versus what the data shows, and the three questions they need answered before building. Written for a non-technical stakeholder.
Technical track	System design: integration patterns, coupling, and failure isolation between systems.
Evidence	Working connector with state persistence, drift detection test, three ingested sources, reconciliation report, source-system brief, documented assumptions and open questions.
Sprint exit	Learner can extract reliably from an undocumented source system they cannot modify, and prove the extract is complete.
Scope guardrail	Read-only access to source systems. Never modify a client source; every integration is additive.

SPRINT 14 · Calendar Week 18

Pipeline Orchestration, Data Quality and Reconciliation

Make pipelines durable, observable and trustworthy — because the client will notice a wrong number long before they notice a slow query.

Self-paced Mon–Fri	– Temporal: workflows, activities, retries, timeouts and durable execution [M] – Idempotency, exactly-once semantics and compensating actions [M] – Scheduling, backfill, replay and partial reprocessing [M] – Great Expectations: expectation suites, checkpoints and validation gates [M] – dbt tests and transformation testing concepts [H] – Reconciliation: row counts, control totals, referential integrity and drift alerts [M] – Dead-letter handling, quarantine tables and operator remediation paths [M] – Pipeline observability: run history, lineage and cost of a failed run [H]
Saturday Implementation Studio	– Live build: the ingestion layer wrapped in a Temporal workflow with retries and timeouts – Backfill and replay executed live against a deliberately corrupted run – Compensating action written for a partially applied load
Sunday Field Engineering	– Technical Clinic — system design: durable execution and failure recovery – Live build: data-quality gates that block a bad load rather than reporting it afterwards – Reconciliation dashboard produced and a discrepancy traced to source in-session – Field Lab and sprint review
Projects	– Implementation Project 27 — Durable Ingestion Workflow. The ingestion layer orchestrated with Temporal: retries, timeouts, backfill, replay, compensating actions and run history. – Implementation Project 28 — Data Quality and Reconciliation Gate. Expectation suites that block bad loads, a quarantine path for rejected records, and a reconciliation report the client can read.
Field Lab (Client Engagement)	– Toolchain: Great Expectations · Metabase · Loom · Claude – Delivering bad news. Learners discover a material data-quality problem in the client's source — and must record a five-minute Loom explaining it to a non-technical stakeholder: what is wrong, what it means for the project, what they recommend, and what they need from the client. Graded on clarity and on not assigning blame.
Technical track	System design: durable workflows, retry semantics and data-integrity guarantees.
Evidence	Temporal workflow with run history, backfill and replay demonstration, compensating action test, expectation suite blocking a bad load, quarantine path, reconciliation report, stakeholder Loom.
Sprint exit	Learner can operate a durable, self-verifying pipeline and explain a data-quality failure to a non-technical stakeholder.
Scope guardrail	One orchestrator implemented properly. Airflow remains an architectural comparison.

SPRINT 15 · Calendar Week 19

Discovery, Scoping and the 72-Hour Prototype

The sprint that makes this an FDE programme. Arrive at a stakeholder with a vague complaint, leave with an agreed scope — then prove value in three days.

Major Milestone 5 falls in this sprint. Capstone client assigned.

Self-paced Mon–Fri	– Stakeholder mapping, interview technique and open questioning [M] – Problem framing: symptom versus cause, and the question behind the request [M] – Requirement elicitation, current-state process mapping and pain quantification [M] – Writing a discovery brief and a solution architecture summary [M] – Scoping and estimating under uncertainty; assumptions and exclusions [M] – Statement of work structure, acceptance criteria and change control [H] – Saying no: scope negotiation, trade-off framing and expectation setting [M] – Executive communication: the one-page summary and the three-slide update [M]
Saturday Implementation Studio	– Live stakeholder interviews with role-playing clients — recorded and critiqued as a group – Requirements workshop facilitated by learners; instructors play a difficult stakeholder – Scope negotiation drill: the client adds a fifth requirement on day two
Sunday Field Engineering	– Technical Clinic — case interview format: framing an ambiguous business problem – 72-hour prototype challenge launches: deliberately scrappy, time-boxed, presented Monday – Discovery brief, architecture summary and estimate reviewed and approved before any build begins – Field Lab, milestone review and capstone client assignment
Projects	– Implementation Project 29 — Discovery Brief and Scoped SOW. From a vague stakeholder complaint: a stakeholder map, current-state process, quantified pain, solution architecture, assumptions, exclusions, estimate and acceptance criteria. – Implementation Project 30 — 72-Hour Prototype and Go/No-Go — Major Milestone 5. A time-boxed working prototype presented to the client for a go/no-go decision, with an explicit list of what was faked and what would need rebuilding.
Field Lab (Client Engagement)	– Toolchain: Miro · Notion · Linear · Mermaid · Metabase · Claude – Scope defence. Each learner presents their SOW to a panel playing procurement, IT security and the business sponsor — three stakeholders with conflicting priorities. The learner must hold scope, surface risk honestly, and leave with a signed-off document. Graded on scope discipline, not on agreeableness.
Technical track	Technical Clinic shifts to case-interview format: ambiguous business problems, structured framing and hypothesis-driven questioning.
Evidence	Stakeholder map, interview notes, current-state process map, discovery brief, solution architecture, estimate with assumptions and exclusions, SOW with acceptance criteria, working prototype, go/no-go presentation, faked-versus-real inventory.
Sprint exit	Learner can run a discovery conversation, produce a defensible scope, and prove value inside a week without over-promising.
Scope guardrail	The prototype is disposable by design. Learners who over-engineer it are marked down; the graded skill is deciding what not to build.

PHASE IV · Enterprise Deployment, AI and Handover

SPRINT 16 · Calendar Week 20

Containers, Kubernetes and Reproducible Environments

Package a system so it runs identically on the learner's laptop, a client's cluster and an air-gapped server.

Self-paced Mon–Fri	– Docker images, layers, multi-stage builds, non-root users and image size [M] – Docker Compose for the full local stack with health checks [M] – Kubernetes objects: pods, deployments, services, ingress, config maps and secrets [M] – Resource requests and limits, liveness and readiness probes, rolling updates [M] – Helm charts, values files and per-client overrides [M] – Private registries, image signing and vulnerability scanning [H] – Air-gapped delivery: offline bundles, image tarballs and installation runbooks [H] – Persistent volumes, stateful workloads and backup considerations [H]
Saturday Implementation Studio	– Live build: the full stack containerised and running under Compose with health checks – Live migration of that stack onto Kubernetes — the same system, a different substrate – Failure drill: kill a pod, watch the deployment recover, then break the readiness probe deliberately
Sunday Field Engineering	– Technical Clinic — system design: deployment topology and blast radius – Live build: a Helm chart with per-client values files producing three different deployments – Air-gapped bundle produced and installed from a runbook, with no internet access – Field Lab and sprint review
Projects	– Implementation Project 31 — Containerised Stack with Health Checks. The full system running under Compose with multi-stage builds, non-root users, health checks and a documented startup path. – Implementation Project 32 — Helm-Deployed Multi-Client Release. A Helm chart deploying the system to Kubernetes with per-client values, probes, resource limits, and an air-gapped installation bundle with a runbook.
Field Lab (Client Engagement)	– Toolchain: Kubernetes · Helm · Claude · Notion – The constrained environment. Learners receive a fictional client's infrastructure constraints — no egress, an old Kubernetes version, a mandated registry, a two-week change window — and must produce a deployment plan that works within them, plus the list of constraints they would push back on and why.
Technical track	System design: deployment topology, blast radius and rollback strategy.
Evidence	Dockerfiles, Compose file, Kubernetes manifests, Helm chart with three values files, probe configuration, pod failure drill, air-gapped bundle, installation runbook, constrained-environment deployment plan.
Sprint exit	Learner can package and deploy a system to a Kubernetes cluster they do not own, including offline.
Scope guardrail	One managed cluster and one air-gapped simulation. Service mesh, operators and multi-cluster federation are out of scope.

SPRINT 17 · Calendar Week 21

Infrastructure as Code, Enterprise Cloud and Secrets

Provision and hand over infrastructure the client can audit, reproduce and take ownership of.

Self-paced Mon–Fri	– Terraform: providers, resources, state, modules and remote backends [M] – Environment separation, workspaces and per-client infrastructure modules [M] – AWS core: VPC, subnets, security groups, IAM, RDS, S3 and managed Kubernetes [M] – Azure equivalents: VNet, Entra ID, Azure Database, Blob Storage and AKS [H] – HashiCorp Vault and cloud secret managers; dynamic credentials and rotation [M] – CI/CD with GitHub Actions: plan, apply, environment gates and rollback [M] – Network constraints: private endpoints, egress rules, proxies and firewall exceptions [H] – Cost estimation, tagging and the infrastructure line in a client proposal [H]
Saturday Implementation Studio	– Live build: Terraform provisioning a client-shaped environment — network, database, cluster, secrets – Plan-and-apply discipline in-session: reading a plan before trusting it – Live drill: destroy and reprovision from state, proving reproducibility
Sunday Field Engineering	– Technical Clinic — system design: environment strategy and infrastructure ownership handover – Live build: the delivery pipeline with environment gates, secret injection and a rollback path – Secrets rotated live without redeploying the application – Field Lab and sprint review
Projects	– Implementation Project 33 — Terraform Client Environment. A reproducible client-shaped environment as code: network, database, cluster, secrets and IAM, with modules, remote state and a destroy-and-rebuild proof. – Implementation Project 34 — Gated Delivery Pipeline. A pipeline running lint, type check, tests, build, security scan, plan, gated apply, smoke test and rollback, with secrets injected at runtime and never committed.
Field Lab (Client Engagement)	– Toolchain: Terraform · Vault · Claude · Notion – The security questionnaire. Learners complete a realistic enterprise security review — data residency, encryption at rest and in transit, access control, logging, subprocessors, incident response — against their own system, and must honestly mark the items they cannot yet satisfy along with a remediation plan and timeline.
Technical track	System design: environment strategy, infrastructure ownership and the handover boundary.
Evidence	Terraform modules, remote state, destroy-and-rebuild proof, pipeline with gates, secret rotation demonstration, rollback evidence, completed security questionnaire with honest gaps and remediation plan.
Sprint exit	Learner can provision, deliver and hand over client infrastructure as auditable code.
Scope guardrail	AWS at mastery, Azure at hands-on. One cloud provisioned end to end beats two configured partially.

SPRINT 18 · Calendar Week 22

AI on Client Data: Private Inference, RAG and Evaluation

Deploy AI where the constraints are real — the client's documents, the client's permissions, the client's regulator.

Major Milestone 6 falls in this sprint.

Self-paced Mon–Fri	– Private inference: Azure OpenAI, AWS Bedrock, VPC endpoints and data-handling terms [M] – Self-hosted inference with vLLM and Ollama; quantization and hardware sizing [H] – Document ingestion with Unstructured.io and Docling; OCR for scanned records [M] – Permission-aware retrieval: per-user, per-tenant and per-document access at query time [M] – Chunking strategy, hybrid search and reranking on real enterprise documents [M] – PII detection and redaction with Presidio; DPDP Act and data-residency obligations [M] – Evaluation with Ragas and Promptfoo: faithfulness, relevance, and a client-signed golden set [M] – Guardrails, refusal behaviour, citation requirements and human review paths [H] – Cost, latency and token budgets in a client proposal [H]
Saturday Implementation Studio	– Live build: a document corpus ingested with OCR, chunked, embedded and made searchable in pgvector – Permission-aware retrieval implemented and then attacked — can user B see user A's document? – PII redaction applied at ingestion and verified
Sunday Field Engineering	– Technical Clinic — case interview: scoping an AI project a client has already over-promised internally – Live build: an evaluation harness on a client-representative golden set, with faithfulness and citation checks – Cost and latency budget produced; the honest answer on what the model cannot do – Field Lab, milestone clinic and sprint review
Projects	– Implementation Project 35 — Permission-Aware RAG Service. Document ingestion with OCR and PII redaction, hybrid retrieval with reranking, per-user and per-tenant access enforced at query time, and citations on every answer. – Implementation Project 36 — AI Evaluation and Business Case — Major Milestone 6. A golden test set signed off by the client stakeholder, faithfulness and relevance scores, failure-mode catalogue, cost and latency budget, and a one-page business case with honest limitations.
Field Lab (AI-Native)	– Toolchain: Ragas · Promptfoo · Presidio · Langfuse · Azure OpenAI / Bedrock · vLLM – Proving value, honestly. Learners must produce the number the client will be judged on — accuracy on their documents, hours saved, cost per query — and defend the measurement method. Any metric that cannot survive a sceptical CFO is rejected.
Technical track	Case interview: scoping an AI project with an unrealistic internal expectation already set.
Evidence	Ingestion pipeline with OCR and redaction, permission-scoped retrieval with an attack test, citation coverage, golden test set, Ragas scores, failure-mode catalogue, cost and latency budget, business case.
Sprint exit	Learner can deploy an evaluated, permission-aware AI feature on client data and defend its value with measurements.
Scope guardrail	No claim of model training or fine-tuning capability. Retrieval, evaluation and deployment only.

SPRINT 19 · Calendar Week 23

Observability, Reliability and Compliance Evidence

Make the system explain itself, so the client's team can operate it without the engineer who built it.

Self-paced Mon–Fri	– OpenTelemetry: traces, metrics, logs and context propagation across services [M] – Prometheus and Grafana: instrumentation, dashboards and alert rules [M] – Loki for log aggregation; structured logging and correlation IDs [M] – Sentry for error tracking, release health and regression detection [M] – PostHog for product analytics, including self-hosted deployment [H] – SLOs, error budgets and alerting that does not cry wolf [M] – Incident response: severity levels, on-call, runbooks and blameless postmortems [M] – Audit logging, log retention and evidence for SOC 2 and ISO 27001 controls [H] – DPDP Act obligations: consent, purpose limitation, retention and data-principal rights [H]
Saturday Implementation Studio	– Live build: distributed tracing across console, API, worker and pipeline with one correlation ID – Dashboards and alerts built for the three things that actually page someone – Live incident: an instructor breaks production, learners diagnose from telemetry alone
Sunday Field Engineering	– Technical Clinic — system design: observability, SLOs and operational ownership – Live build: audit logging and retention policy implemented against a compliance control list – Postmortem written in-session for the morning's incident – Field Lab and sprint review
Projects	– Implementation Project 37 — Instrumented Distributed System. End-to-end tracing, metrics, structured logs and error tracking across every service, with dashboards and alerts tied to defined SLOs. – Implementation Project 38 — Incident Response and Compliance Pack. A rehearsed incident with a blameless postmortem, plus audit logging, retention policy and a control-mapping document covering SOC 2 and DPDP obligations.
Field Lab (Client Engagement)	– Toolchain: Grafana · Sentry · PostHog · Notion · Loom – The operational handover rehearsal. Learners run a live session teaching a fictional client operations team to use the dashboards, interpret an alert and follow the runbook — then the team is given a failure to resolve without help. The learner is graded on whether the team succeeded, not on the quality of the explanation.
Technical track	System design: observability strategy, SLO definition and operational ownership transfer.
Evidence	Trace across four services, dashboards, alert rules with SLO rationale, error tracking, incident timeline, blameless postmortem, audit log, retention policy, control-mapping document, handover rehearsal result.
Sprint exit	Learner can instrument a distributed system, respond to an incident from telemetry, and produce compliance evidence a client auditor will accept.
Scope guardrail	Compliance is evidence production and control mapping, not certification. No claim of being an auditor.

SPRINT 20 · Calendar Week 24

Handover, Enablement and Client Capstone Defence

Finish the engagement the way an FDE is measured: the client can run it, extend it and no longer needs you.

Major Milestone 7 falls in this sprint. Programme completion.

Self-paced Mon–Fri	– Release readiness: checklist, defect triage, go/no-go decision and known-issues register [M] – Runbook authoring: routine operations, failure modes, escalation and contacts [M] – Administrator and end-user documentation; training material design [M] – Enablement session design: teaching a team to operate what you built [M] – Support model, SLAs, escalation paths and the first-90-days plan [H] – Quarterly business review structure: outcomes, adoption, roadmap and renewal [H] – Writing field observations back into product requirements [H] – Pilot-to-production conversion: the commercial conversation an engineer is in the room for [H] – Case-study and portfolio writing; technical interview narrative [M]
Saturday Implementation Studio	– Release checklist executed live; defect triage and go/no-go made as a group – Runbook and admin guide written in-session and stress-tested by a peer who has never seen the system – Highest-risk gaps closed with instructor pairing
Sunday Field Engineering	– Technical Clinic — final case interview and system-design panel with written feedback – Client capstone defence: live demonstration and technical defence before a panel playing sponsor, IT and security – Handover pack delivered; QBR deck presented – Final portfolio, placement plan and programme close
Projects	– Implementation Project 39 — Handover Pack and Enablement Session. Runbook, administrator guide, end-user documentation, support model and a delivered training session — validated by having someone else operate the system unaided. – Implementation Project 40 — Client Capstone and Defence — Major Milestone 7. The deployed client system defended before a panel: discovery, architecture, integration, deployment, AI evaluation, operations, limitations and commercial outcome.
Field Lab (Client Engagement)	– Toolchain: Loom · Metabase · Notion · Claude · full toolchain – The QBR. Learners present a quarterly business review to a panel playing the client sponsor: what was delivered, what it is worth, what did not work, what is next and what it will cost. Graded on honesty about failure as much as on the outcome.
Technical track	Final case interview and system-design panel. Technical assessment complete.
Evidence	Release checklist, defect register, deployed client system, runbook, admin guide, training session recording, unaided-operation proof, support model, QBR deck, capstone defence, case study, portfolio.
Sprint exit	Learner has a deployed client system, a validated handover, and evidence for forward deployed and deployment engineering roles.
Scope guardrail	Career claims must match demonstrated evidence. The outcome is a credible associate forward deployed engineer, not a guaranteed placement.

11. The Client Engagement

Every learner is assigned a simulated client at the end of Calendar Week 19 and carries that engagement to graduation. The client is played by an instructor or an alumnus with a defined organisation, a defined problem, competing internal priorities and a personality. Requirements change. Access is granted late. Someone in IT security asks a question nobody prepared for. This is deliberate.

Week	Engagement stage	What the learner does	Deliverable
19	Discovery and scoping	Stakeholder interviews, current-state process mapping, pain quantification, solution architecture, estimation under uncertainty.	Discovery brief, solution architecture, estimate with assumptions and exclusions, scoped SOW with acceptance criteria.
19	72-hour prototype	Time-boxed build proving the core hypothesis, presented for a go/no-go decision.	Working prototype, go/no-go presentation, explicit inventory of what was faked.
20–21	Deployment groundwork	Environment provisioning within the client's stated constraints; identity federation; pipeline and secrets.	Terraform environment, Helm deployment, completed security questionnaire with honest gaps.
22	AI capability	Document ingestion, permission-aware retrieval, evaluation against a golden set the client signs off.	Evaluated AI feature, failure-mode catalogue, cost and latency budget, one-page business case.
23	Operational readiness	Instrumentation, SLOs, alerting, an incident rehearsal, audit logging and control mapping.	Dashboards, alert rules, blameless postmortem, compliance pack.
24	Handover and close	Runbook, administrator guide, enablement session, support model, quarterly business review.	Handover pack, proof of unaided operation by another party, QBR deck, capstone defence.

The scope pressure is the curriculum

At three points in the engagement the client introduces a change the learner did not plan for: a new requirement in week 21, an infrastructure constraint in week 22, and a request to expand scope without expanding the timeline in week 23. Learners are not told when these will arrive. How they respond is assessed under the Discovery, Scoping and Client Communication rubric — and a learner who quietly absorbs all three scores lower than one who pushes back with a clear trade-off.

12. Major Milestone Ladder

Every sprint ends with a project. The seven milestones below receive a deeper portfolio review; the remaining implementation projects provide progressive evidence between them.

#	Sprint	Project	Primary evidence	Portfolio value
1	04	Typed API Application	Strict TypeScript, runtime validation, network resilience, tests	Language and typing evidence
2	06	Tested Accessible Console	React architecture, state ownership, testing, accessibility, performance, architecture decision record	Frontend completion gate
3	08	Persistent Data API	PostgreSQL modelling, migrations, indexes, query plans, transactional writes, integration tests	Backend and data evidence
4	11	Background Jobs and Real-Time Feature	Caching policy, idempotency, queued work, dead-letter handling, real-time feature, failure tests	Production workflow evidence
5	15	Discovery, SOW and 72-Hour Prototype	Stakeholder map, discovery brief, architecture, estimate, SOW, prototype, go/no-go	The distinctively FDE artifact
6	18	AI Evaluation and Business Case	Permission-aware RAG, golden set, faithfulness scores, failure catalogue, cost budget, business case	AI deployment evidence
7	20	Client Capstone and Defence	Deployed system, handover pack, unaided-operation proof, QBR deck, panel defence	Graduate showcase

13. Placement Track

Placement begins in Calendar Week 13, after the frontend gate and alongside backend work. Target roles are deployment engineer, solutions engineer, implementation engineer, integration engineer and associate forward deployed engineer. The track consumes roughly three hours per week and is embedded in the timetable.

Period	Technical position	Placement activity	Required evidence
Weeks 13–15	Frontend and backend portfolio; enterprise identity added	Resume and profile alignment; applications to implementation and solutions engineering roles; system-design practice	Approved resume, portfolio links, application tracker, one mock system-design interview
Weeks 16–18	Integration capability established	Recruiter conversations; take-home practice on data problems; integration architecture questions	Two application reviews, one timed data task, technical story bank
Weeks 19–21	Discovery artifacts and deployment capability	Expansion to deployment and forward deployed roles; case-interview practice; employer events	SOW as portfolio artifact, deployment demo, second mock interview
Weeks 22–23	AI on client data; operational maturity	Networking, referral activity, AI project scoping questions, business-case	Capstone preview, business case, interview feedback actions

Period	Technical position	Placement activity	Required evidence
		articulation	
Weeks 24	Complete client engagement	Showcase, final panel rehearsal, application follow-up, consulting-readiness practice	Capstone defence, case study, final profile and placement plan

14. Assessment and Certification

Assessment is continuous and evidence-based. There is no single final examination; the composite score is assembled from work produced throughout the programme. The weighting differs from a standard engineering programme: the client engagement and the field skills carry more, because that is what the role is.

Assessment component	Weight	Evidence
Client capstone and defence	25%	Deployed client system, architecture, integration, operations, handover pack, unaided-operation proof, panel defence.
Major milestone projects	20%	Seven deeper submissions assessed for functionality, code quality, security, testing and delivery.
Implementation projects	15%	Forty scoped builds showing continuous application, debugging and improvement across the sprints.
Discovery, scoping and client communication	15%	Nine Client Engagement Labs, the discovery brief and SOW, recorded stakeholder interactions, and response to mid-engagement scope pressure.
Integration reliability and deployment	10%	Connector state and drift handling, reconciliation, durable workflows, Helm and Terraform reproducibility, security questionnaire honesty.
Technical assessment	10%	Three checkpoints across DSA Levels 1–3, system design and case interviews, plus weekly problem submissions.
Live studio participation and self-paced completion	5%	Twenty sprint demonstrations, pull requests, review responses, readiness credit, platform module completion and weekly quizzes.

Certification thresholds

Requirement	Threshold
Composite score	70 / 100 minimum
Client capstone score	6 / 10 minimum, with no individual capstone dimension below 5 / 10
Discovery, scoping and client communication	6 / 10 minimum — a technically excellent learner who cannot hold a client conversation is not certified
Live studio attendance	80% minimum across the forty-eight sessions
Technical checkpoints	All three attempted; Checkpoint 2 passed before placement approval
Milestone submissions	All seven submitted; Milestone 2 passed before placement approval
Handover validation	Another party must operate the capstone system unaided from the runbook

Minimum completion evidence

- Forty implementation project demonstrations or approved make-up submissions.

- Twenty sprint review records and seven major milestone submissions.
- At least one deployed internal console, one documented API, one multi-source ingestion layer and one client system deployed to Kubernetes.
- A custom connector with persisted state and a drift-detection test, plus a reconciliation report proving extract completeness.
- A Terraform environment with a destroy-and-rebuild proof, and a Helm chart producing at least three per-client deployments.
- An enterprise SSO integration over OIDC with SAML fallback, and an automated cross-tenant leakage test suite.
- An evaluated AI feature with permission-aware retrieval, a client-signed golden set and a failure-mode catalogue.
- Distributed tracing across four services, an incident timeline, a blameless postmortem and a control-mapping document.
- A discovery brief, scoped SOW, and a completed security questionnaire with honestly marked gaps.
- A handover pack validated by having another party operate the system unaided.
- A complete AI Disclosure Log covering the full twenty-four weeks, including rejected suggestions and verification method.
- Nine recorded Client Engagement Lab sessions with written self-critique.

15. Who This Programme Is For

A strong fit

- Engineers with two or more years of professional experience who want to move from product work into deployment, solutions or forward deployed engineering.
- Graduates of the Impacteers AI-Native Full-Stack Engineer Program ready to specialise.
- Backend or full-stack engineers who already integrate with third-party systems and want to do it at enterprise scale, with the client conversation included.
- Consultants and solutions engineers with strong client skills who need the deployment and data engineering depth to match.
- Engineers who are energised rather than drained by ambiguity, unfamiliar systems and stakeholders who change their minds.
- Anyone who can protect Saturday and Sunday mornings for six months and give eleven to sixteen hours a week.

Not a fit

- Complete beginners. This programme assumes prior professional engineering experience or completion of the full-stack programme; it moves at a pace that will not accommodate learning to code.
- Anyone seeking a guaranteed placement. Forward deployed roles are competitive and often expect more experience than a six-month programme can manufacture.
- Engineers who want to stay away from stakeholders. Half the differentiating content of this programme involves a person on the other side of a conversation, live and recorded.
- Anyone looking for depth in model training, fine-tuning, semiconductor design or MLOps research. AI here means deployment, retrieval and evaluation on client data.
- Learners who want a passive video course. Six live hours a week with camera on, code shared and client role-plays recorded is the core of the delivery model.
- Anyone who cannot commit eleven to sixteen hours per week, rising during the client engagement.

Impacteers · The AI Forward Deployed Engineer Program · Cohort 01

Client engagements in this programme are simulated with role-playing stakeholders, and this is stated openly in all learner-facing and employer-facing material. Salary, demand and hiring references used in supporting materials are drawn from India 2026 market guides and are indicative rather than guaranteed. Curriculum content, tooling and depth codes are reviewed each cohort and may be updated to reflect current industry practice.

impacteers.com