IMPACTEERS

AI-NATIVE ENGINEERING CERTIFICATION · COHORT 01

The AI-Native Full-Stack Engineer Program

MERN Engineering · AI Engineering · Interview-Ready DSA

A six-month, weekend-live career program that builds production MERN engineers who work fluently alongside Claude, Cursor and the modern AI toolchain — and who can prove it with shipped, tested, defended software.

24	18	36	7	110+
CALENDAR WEEKS	SPRINTS	PROJECTS	MILESTONES	DSA PROBLEMS

144	96	240	W13
LIVE HOURS (SAT & SUN)	SELF-PACED HOURS	TOTAL STRUCTURED HOURS	PLACEMENT BEGINS

Programme prospectus and curriculum specification · impacteers.com

Contents

1. Programme Positioning

Three shifts have changed what a junior full-stack hire is expected to be. This programme is built around all three rather than treating any of them as an optional add-on.

The shift	What it means for this programme
AI tool fluency is now baseline	Writing code with an assistant is no longer a differentiator; using one well is. Learners work with Claude, Claude Code, Cursor, Copilot and the MCP ecosystem from Sprint 01, under a disclosure and verification discipline that hiring managers can inspect.
Judgment matters more than output	When generation is cheap, the scarce skill is deciding what to keep. Every AI-Native Lab in this programme requires learners to reject something and defend the rejection. Assessment rewards the review, not the volume.
Production discipline separates candidates	Tests, observability, security boundaries, rollback plans and honest limitation notes are what distinguish a hireable junior from a tutorial graduate. Every sprint ends with evidence, not with a completed video.

What this programme claims

- Junior full-stack engineering readiness across the MERN stack with TypeScript throughout.
- Working, supervised fluency with the modern AI development toolchain and with AI application patterns — RAG, tool calling, MCP, agents and evaluation.
- Interview-ready data structures and algorithms capability, assessed five times and rehearsed under panel conditions.
- A defensible portfolio: a deployed frontend capstone, a documented API, and a production-style full-stack capstone with operational evidence.

What this programme does not claim

- Guaranteed employment. Career claims must match demonstrated evidence.
- Mastery of foundation-model training, semiconductor design, robotics hardware or autonomous-vehicle operation. These appear at stated exposure depth to build systems literacy.
- Senior or architect-level production experience. The honest outcome is a well-evidenced junior engineer.

2. The Learning Model

The programme runs on a strict two-layer split. Weekdays are for acquiring language, syntax and API 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 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 API surface – Guided setup and configuration – Browser-IDE coding practice – Weekly quiz and debugging challenge – DSA problem sets with written complexity notes – Cheat sheets and pre-session readiness check	– Implementation of the sprint's projects, built live – Architecture decisions argued in the open – Debugging real failures the instructor introduces – Trade-off discussion and defence – Code review, pull requests and demos – AI-Native Labs on the tool of the sprint – DSA Live Clinic: pattern teaching and timed 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
Format	Video, notes, interactive coding, quizzes, problem sets — all on the Impacteers platform, available throughout the programme.	Instructor-led, camera-on, screen-shared. Recorded and published to the platform 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. The self-paced layer is the entry ticket, not a supplement.

Time commitment

Component	Per week	Notes
Self-paced platform learning	~4 hours	Mon–Fri, flexible. Concept videos 1.25 hrs, interactive coding 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.
Sunday Applied Engineering Studio	3 hours	Live. DSA clinic, sprint project, AI-Native Lab and sprint review.
Independent project work	3–5 hours	Completing the sprint project, tests and evidence between sessions.
Placement track (from Week 13)	~3 hours	Applications, mock interviews, recruiter conversations, profile work.
Total	10–15 hours	Rising to 13–18 hours from Calendar Week 13 when placement activity begins.

3. The Weekly Rhythm

Every week of the programme follows the same shape. The predictability is deliberate: learners plan around it, and instructors reuse the structure so that studio time is spent on engineering rather than on 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 the language rules, syntax and API surface for the sprint.
Interactive coding practice	1 hr 00	Browser IDE drills. Fluency work — typing the syntax 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 snippet to diagnose. Contributes to the grade.
DSA problem set	ongoing	Five problems per week from Calendar Week 3, each submitted with a written complexity justification.
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 along in their own repository.
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 code becomes theirs.
2:40 – 3:00	Debug clinic and brief	Defects introduced deliberately and diagnosed as a group. Sunday's scope briefed.

Sunday · Applied Engineering Studio · 3 hours live

Clock	Segment	Purpose
0:00 – 0:45	DSA Live Clinic	Pattern teaching, one timed problem solved under interview conditions, then a full solution walkthrough with complexity analysis.

Clock	Segment	Purpose
		Runs from Calendar Week 3 to graduation.
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	AI-Native Lab	The tool or pattern of the sprint, applied to the learner's own codebase. Every lab requires a rejection with written reasoning.
2:35 – 3:00	Sprint review	Demo, pull request review, evidence check and retrospective. Milestone sprints extend this segment.

Why implementation is reserved for live time

Syntax can be learned alone; judgment cannot. The decisions that separate a working application from a defensible one — where state lives, what the error boundary swallows, whether an index is worth its write cost, which AI suggestion to throw away — are learned by watching them argued and then arguing them. That is what the six weekend hours are protected for.

4. The AI Toolchain

Learners do not merely hear about AI tooling; they work in it every sprint. The toolchain below is taught in sequence, and each tool carries a depth code indicating expected independence. Tools are introduced when a project needs them, never as a standalone module.

AI coding assistants and IDEs

Tool	Depth	How it is used in the programme
Claude	M	Primary reasoning, explanation and code-review assistant. Introduced Sprint 01 as an explainer rather than a generator, configured as a Socratic hint-ladder tutor for DSA from Sprint 03, and used for architecture critique through the capstone.
Claude Code	M	Agentic, repository-aware coding from the terminal. Used for large refactors, test-suite generation, codebase questions and documentation. Introduced Sprint 08 for AI-generated tests; central to Sprints 10, 11, 17 and 18.
Cursor	M	AI-native IDE. Inline edits, multi-file Composer changes and codebase-wide context. Introduced Sprint 01; Composer workflows taught in Sprint 06 with a mandatory structured review pass.
GitHub Copilot	H	Inline completion and pull-request summaries. Deliberately compared against hand-written implementations in Sprint 03 so learners can feel where completion helps and where it drifts.
Windsurf, Cline, Continue.dev	E	Alternative agentic editors demonstrated so learners can evaluate the category rather than depending on one vendor.

Model providers and runtimes

Tool	Depth	How it is used in the programme
Anthropic API (Claude models)	M	Primary application-layer model API. Typed clients, structured output, streaming and error handling built from Sprint 05 onward.
OpenAI API	H	Second provider, used to build and test the provider-abstraction layer in Sprint 10 so no application is locked to one vendor.
Google Gemini API	E	Comparative exposure — multimodal input and differing structured-output behaviour.
Ollama and LM Studio	H	Local inference. Learners run quantized open models on their own machines in Sprint 16 to make compute, memory and latency trade-offs concrete rather than theoretical.
Hugging Face	E	Model discovery, licensing review and inference endpoints.

AI application frameworks and patterns

Tool	Depth	How it is used in the programme
Vercel AI SDK	H	Streaming responses, structured output and conversation state inside React. Introduced

Tool	Depth	How it is used in the programme
		Sprint 07 for AI interaction UX.
Zod	M	Runtime schema validation at every boundary, including model output. The core defence against trusting a generated response.
Model Context Protocol (MCP)	H	Tool definition with typed schemas, permissions, validation and audit logs. Conceptual seed in Sprint 04, implemented from Sprint 10, governed with a permission matrix in Sprint 17.
LangChain.js and LlamaIndex.TS	H	Retrieval chains, document loaders and agent scaffolding for the RAG service in Sprint 12.
BullMQ and Redis	H	Background agent execution with queues, retries and dead-letter handling in Sprint 14.
n8n	E	Visual workflow orchestration, demonstrated as the low-code counterpart to a coded agent.

Retrieval, vector storage and evaluation

Tool	Depth	How it is used in the programme
MongoDB Atlas Vector Search	M	Primary vector store, chosen so retrieval stays inside the MERN stack learners already operate. Ingestion, metadata filtering and hybrid search built in Sprint 12.
Pinecone	H	Managed vector database, used to compare hosted retrieval against in-database retrieval.
Chroma and Qdrant	E	Local and open-source alternatives, demonstrated in lab.
Embedding models	H	Chunking strategy, dimensionality, cost per million tokens and the effect of chunk size on retrieval quality.
Promptfoo	H	Golden test sets and regression evaluation for AI features. Introduced Sprint 08 alongside conventional testing.
Langfuse and LangSmith	H	Tracing, prompt-version tracking, latency and cost observability. Wired into the capstone in Sprint 17.

AI-assisted quality, review and security

Tool	Depth	How it is used in the programme
Vitest, Jest and Playwright	M	Conventional test tooling. AI may scaffold tests; learners must delete every test that passes for the wrong reason and add what the model missed.
CodeRabbit and AI review workflows	H	Automated pull-request review compared against human review, with learners cataloguing what each catches and misses.
Snyk and Dependabot	E	Dependency and vulnerability scanning inside the CI pipeline.
Perplexity and Claude with search	H	Specification research and library evaluation, with source verification required before any dependency is adopted.

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.

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.

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 interfaces using semantic HTML, modern CSS, React and TypeScript.
- Select appropriate approaches for local UI, form, URL, server-cache and global client state.
- Design typed REST and real-time services using Node.js, Express, MongoDB, Redis and WebSockets.
- Implement secure identity, session or token, authorization and third-party integration workflows.
- Write unit, component, integration, contract and end-to-end tests throughout development.
- Containerize applications, automate delivery, deploy to cloud platforms, and operate with logs, health checks, backups and rollback plans.
- Compare relational and document data models and justify a storage decision with access patterns.

AI engineering outcomes

- Build and evaluate Generative AI features using foundation-model APIs, structured prompting and explicit verification.
- Design agentic workflows that plan, call tools, maintain bounded state, recover from failure and keep humans in control of consequential actions.
- Implement retrieval-augmented generation with chunking, embeddings, vector search, metadata filtering, citations and retrieval evaluation.
- Expose and consume tools through MCP with explicit schemas, permissions, validation and audit logging.
- Apply AI engineering practice: model selection, evaluation harnesses, observability, security, cost control, latency budgets and fallbacks.
- Explain AI compute fundamentals — CPU, GPU, NPU and TPU roles, inference versus training, memory bandwidth, quantization and cost trade-offs.
- Design edge AI solutions accounting for device constraints, local inference, privacy, offline behaviour and synchronisation.
- Prototype embodied and autonomous-system architectures with perception-action loops, telemetry, operational boundaries and fail-safe design.
- Use AI for development responsibly, with review, verification, testing, disclosure and documented rejection.

DSA and career outcomes

- Progress from fundamental problem solving to advanced patterns, complexity analysis and timed coding practice across five assessed levels.
- Solve unseen medium-difficulty problems under interview time pressure and justify the complexity of the chosen approach.
- Present credible frontend and full-stack portfolios with deployed applications, tests and written case studies.
- Defend architecture, trade-offs, limitations, security boundaries and AI usage in a live technical panel.

7. The 24-Week Curriculum Map

Thirty-six learning blocks are delivered across twenty-four calendar weeks as eighteen sprints. Twelve sprints run in a single calendar week; six deeper sprints run across two. No curriculum topic, project, evidence requirement, milestone, DSA level or assessment area has been removed to achieve the pacing.

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 1–12 · Frontend Engineering and Placement Readiness

Sprint	Weeks	Focus	Implementation project outputs
01	1	Web foundations and Git	Semantic profile page; accessible portfolio foundation
02	2	Responsive UI, design systems and UI libraries	Responsive dashboard; reusable UI system · Milestone 1
03	3–4	JavaScript foundations and problem solving	Logic toolkit; DOM task board · DSA Level 1 opens
04	5	Browser applications and async JavaScript	Persistent browser app; public API explorer
05	6	TypeScript and typed API integration	Typed data toolkit; typed API application · Milestone 2
06	7–8	React foundations with TypeScript	Component gallery; React product browser
07	9	Forms, routing and data flow	Validated workflow; routed frontend application
08	10	State, testing, accessibility and performance	State refactor; tested accessible dashboard
09	11–12	Frontend capstone and placement launch	Capstone alpha; deployed frontend case study · Milestone 3

PHASE II · Calendar Weeks 13–20 · Backend Engineering and MERN Integration

Sprint	Weeks	Focus	Implementation project outputs
10	13	Node.js and Express fundamentals	Node service; typed REST API · Placement begins
11	14	API architecture, documentation and testing	Layered refactor; documented tested API
12	15–16	Database foundations, MongoDB and Mongoose	Access-pattern prototype; persistent data API · Milestone 4
13	17	Authentication, authorization and security	Account lifecycle; secure multi-user API
14	18	Production workflows, Redis and real-time	Upload/email/job workflow; caching or real-time feature
15	19–20	Full-stack MERN integration and quality	Integrated user journey; MERN beta release ·

Sprint	Weeks	Focus	Implementation project outputs
			Milestone 5

PHASE III · Calendar Weeks 21–24 · Production Delivery, Capstone and Career Acceleration

Sprint	Weeks	Focus	Implementation project outputs
16	21	Containers, CI/CD, cloud, capstone discovery	Dockerized stack; deployed release and capstone plan
17	22	Capstone vertical slice, observability, responsible AI	Vertical slice; operational beta and AI audit · Milestone 6
18	23–24	Capstone completion and career accelerator	Release candidate; industry capstone presentation · Milestone 7

8. The DSA Track

Data structures and algorithms run in parallel with the engineering curriculum from Calendar Week 3 to graduation. The track is not a bolt-on: it occupies the first forty-five minutes of every Sunday studio, generates five problems per week of self-paced work, and is assessed five times.

Where it lives	Time	What happens	Output
Self-paced · Mon–Fri	~2 hrs	Five problems per week drawn from the current level's patterns, with editorial review after submission.	Five solutions, each with a written complexity justification.
Sunday DSA Live Clinic	45 min	Pattern teaching, one timed problem under interview conditions, then a full solution walkthrough comparing learner approaches.	Live-solved problem and pattern notes.
Level checkpoints	60–90 min	Timed assessment at the close of each level. Five across the programme.	Graded checkpoint contributing to the DSA component.
Mock interviews	45 min each	Panel-conducted coding interviews in Sprints 17 and 18 with written feedback.	Three mock interviews with structured feedback.

The five levels

Level	Weeks	Patterns and depth	Problems	Checkpoint
Level 1 Foundations	3–5	Programming logic and Big-O basics; arrays, strings, objects and hash maps; frequency counting [M]. Sets, linear search and basic sorting [H].	15	End of W5
Level 2 Core patterns	6–9	Two pointers, sliding window and binary search [M]. Prefix sums, stack, queue, linked list and sorting patterns [H].	20	End of W9
Level 3 Recursion & trees	10–13	Recursion, trees, BFS and DFS [H]. Divide-and-conquer intuition, BST, heap and priority queue [H].	20	End of W13
Level 4 Graphs & search	14–18	Graph representation and traversal, backtracking, greedy and intervals [H]. Topological sort and union-find [E].	25	End of W18
Level 5 DP & interview	19–24	Dynamic programming [H/E]. Mixed patterns, timed problems and mock coding interviews [M].	30	End of W24

DSA alignment with the placement track

Period	DSA expectation
Weeks 13–14	Level 3 complete; interview revision across arrays, strings, hash maps and trees.
Weeks 15–17	Level 4 in progress; BFS/DFS, graphs and backtracking under timed conditions.
Weeks 18–20	Level 4 mastery; mixed medium problems at interview pace.

Period	DSA expectation
Weeks 21–22	Level 5 and dynamic programming fundamentals; timed interview sets.
Weeks 23–24	Panel mock coding interviews and revision of high-frequency patterns.

How AI is used — and not used — in DSA

Claude is configured as a Socratic tutor for this track: it gives hint ladders, critiques complexity analysis and generates variant problems. It does not supply solutions. Learners submit their own reasoning first; the assistant is used afterwards to find the gap. Checkpoints and mock interviews are conducted without any assistant, because that is the condition the hiring interview will impose.

9. Sprint-by-Sprint Specification

The full specification for all eighteen 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 AI-Native Lab, the DSA position, the required evidence, the exit condition and the scope guardrail.

PHASE I · Frontend Engineering & Placement Readiness

SPRINT 01 · **Calendar Week 1**

Web Foundations and Professional Git Habits

Understand how the web works, create semantic pages, use browser tooling, and establish reliable repository habits before visual complexity is introduced.

Self-paced Mon–Fri	– Request–response lifecycle, URLs, DNS, hosting [H] – Semantic document landmarks, headings, media, tables and forms [M] – Developer tools, command line, and local environment setup [H] – Cascade, specificity, box model, typography scales [M] – Flexbox, Grid and mobile-first media queries [M] – Git init, commits, branches, pull requests, README authoring [M] – AI and GenAI fundamentals: LLM basics, prompting, limitations, privacy, verification [H]
Saturday Implementation Studio	– Readiness check on the week's self-paced syntax; blocker clinic – Live build: semantic profile page from a blank repository, narrated decision by decision – Browser DevTools walkthrough — inspecting the DOM, network panel, accessibility tree – Learner build sprint with instructor circulation; keyboard-navigation audit
Sunday Applied Engineering	– Complexity primer: how engineers talk about cost before DSA Level 1 opens in Sprint 03 – Live build: extend the profile page into a responsive multi-section portfolio – Git collaboration live — branch, commit hygiene, pull request, review response, merge – Deploy to static hosting in-session; verify the live URL together – AI-Native Lab and sprint review
Projects	– Implementation Project 1 — Semantic Profile Page. A single-page professional profile with meaningful document structure, accessible navigation, media and a validated contact form. – Implementation Project 2 — Accessible Portfolio Foundation. Extend Project 1 into a responsive multi-section portfolio foundation published through static hosting.
AI-Native Lab	– Toolchain: Claude · Claude Code · Cursor – Prompt-and-critique drill. Learners ask Claude to explain a semantic markup decision, then verify the answer against MDN and record where the model was right, vague or wrong. First entry in the AI Disclosure Log.
DSA	Not yet open. Pre-track primer only: reading Big-O notation and recognising cost language in documentation.
Evidence	Repository, semantic audit notes, DevTools inspection screenshot, responsive layout evidence, keyboard review, deployment link, one pull request with a review response, AI disclosure entry.
Sprint exit	Learner can explain the browser request lifecycle, build a semantic responsive page, and deliver it through a basic Git workflow.

Scope guardrail	HTML, CSS, browser tools and Git only. No frameworks, no design-system tooling, no AI-generated markup accepted without line-by-line explanation.

SPRINT 02 · Calendar Week 2

Responsive Interface Engineering and Design Systems

Create polished interfaces from reusable layout rules and visual tokens while holding accessibility and performance baselines.

Major Milestone 1 falls in this sprint.

Self-paced Mon–Fri	– Intrinsic sizing, min/max/clamp, container queries [H] – Advanced Grid, overflow handling, navigation, card, table, form, empty and loading layouts [M] – Typography and spacing scales, colour roles, CSS custom properties, reusable states [M] – Tailwind configuration and utility composition [H] – MUI component library and theming; MUI X overview [H / E] – Image and font performance fundamentals [H]
Saturday Implementation Studio	– Readiness check; live build of a dashboard shell that adapts across mobile, tablet and desktop – Content-stress testing in-session — short, long, empty and error states – Learner build sprint: viewport test matrix produced live
Sunday Applied Engineering	– Live build: design tokens to Tailwind theme, then the same system expressed through MUI – Accessibility review: contrast, focus order, keyboard traps — fixed live – One measured performance improvement, before and after – AI-Native Lab, milestone submission clinic and sprint review
Projects	– Implementation Project 3 — Responsive Operations Dashboard. A dashboard shell that adapts across breakpoints and remains stable with short, long, empty and error content. – Implementation Project 4 — Reusable UI System and Marketing Site — Major Milestone 1. A small reusable interface system applied to a polished responsive site, deployed and documented.
AI-Native Lab	– Toolchain: Claude · Cursor · v0 – AI-assisted UI ideation with human verification. Generate three layout directions, then reject two with written reasoning. Learners must identify one accessibility defect the model introduced.
DSA	Not yet open.
Evidence	Viewport test matrix, documented tokens or Tailwind theme, reusable components, accessibility review, performance measurement, live deployment, AI ideation log with rejections.
Sprint exit	Learner can design and implement a consistent responsive interface system rather than isolated demo pages.
Scope guardrail	Limit the system to components the project actually needs. No general-purpose component library.

SPRINT 03 · Calendar Weeks 3–4

JavaScript Foundations and Problem Solving

Develop reliable programming habits with functions, collections, debugging and small data-structure patterns before framework work begins.

Self-paced Mon–Fri	– let/const, primitives, operators, conditions, loops, functions, scope [M] – Closures, pure functions and side effects [H / M] – Breakpoints, watch expressions, test cases, debugging method [M] – map / filter / find / reduce, object updates, destructuring [M] – Map and Set; local storage [H] – DOM creation, traversal and event handling [M] – Stacks, queues, frequency maps; Big-O intuition [H]
Saturday Implementation Studio	– Live build: JavaScript logic toolkit — validation, pricing, filtering, formatting, rule engines – Debugging clinic — instructor introduces defects live, learners locate them under time pressure – Test-case authoring for every utility written
Sunday Applied Engineering	– DSA Live Clinic opens — Level 1: complexity analysis, arrays, strings, hash maps – Live build: interactive DOM task board with filters, persistence, empty and error states – Keyboard-accessible controls implemented in-session – AI-Native Lab and sprint review
Projects	– Implementation Project 5 — JavaScript Logic Toolkit. Small utilities for validation, pricing, filtering, formatting and rule-based decisions with clear inputs and outputs. – Implementation Project 6 — DOM Task Board. An interactive task board with filters, persistence, empty and error states, and keyboard-friendly controls.
AI-Native Lab	– Toolchain: Claude (Socratic mode) · Cursor · GitHub Copilot – AI-assisted code generation, explanation and debugging. Learners configure Claude as a hint-ladder tutor rather than an answer engine, and compare Copilot inline completion against a hand-written implementation on correctness and readability.
DSA	DSA Level 1 opens [M]: Big-O basics; arrays, strings, objects and hash maps; frequency counting. Sets, linear search, basic sorting [H]. Five async problems per week with written complexity justification; 45-minute live clinic each Sunday.
Evidence	Modular functions, test cases, debugging record, working DOM application, persistence, reusable functions, test checklist, pull request, five DSA solutions with complexity notes.
Sprint exit	Learner can translate requirements into small functions, manipulate application data, and build an interactive browser application without a framework.
Scope guardrail	Practical interview patterns only. Recursion and advanced algorithms remain awareness topics at this stage.

SPRINT 04 · Calendar Week 5

Browser Applications and Asynchronous JavaScript

Organise browser applications into modules, reason about asynchronous execution, and handle real network and UI states.

Self-paced Mon–Fri	– ES modules, imports and exports; npm scripts [M / H] – Feature boundaries, services and utilities; classes versus composition [H] – URL parameters, browser history, storage [H] – Event loop mental model; Promises, async/await, sequencing, try/catch [H / M] – Parallel work, timeouts, retries, cancellation concepts [H / E] – HTTP methods and status codes; loading, empty, error and retry states [M]
Saturday Implementation Studio	– Live refactor: the DOM application becomes a modular application with feature boundaries – Package scripts and clean project setup built in-session – Recoverable UI state design — what the user sees when things fail
Sunday Applied Engineering	– DSA Live Clinic — Level 1 continued: hash-map patterns under time pressure – Live build: public API explorer with cancellation, pagination and normalised errors – Network debugging live — throttling, offline, 500s, malformed payloads – AI-Native Lab and sprint review
Projects	– Implementation Project 7 — Persistent Browser Planner. Refactor the DOM application into modules with URL or local persistence, feature boundaries and recoverable UI states. – Implementation Project 8 — Public API Explorer. A deployed browser application that explores a public API with cancellation, pagination where appropriate, and normalised errors.
AI-Native Lab	– Toolchain: Claude · Cursor · Anthropic API (Messages) – Tool-calling mental model. Learners build a mock function-calling loop in plain JavaScript — schema in, validated arguments out — then wire a resilient AI API client with loading, timeout, retry and cancellation behaviour. This is the conceptual seed for MCP work in Sprint 10.
DSA	DSA Level 1 completes [M]. Checkpoint 1 assessed at the end of Calendar Week 5: timed array, string and hash-map set with complexity justification.
Evidence	Module map, package scripts, persistence test, API client, loading/error/empty states, network debugging notes, deployment, tests for core logic, DSA Checkpoint 1 result.
Sprint exit	Learner can build a modular asynchronous browser application and explain common network failure states.
Scope guardrail	One public API and one clear user journey. Advanced caching and authentication are deferred.

SPRINT 05 · Calendar Week 6

TypeScript and Typed API Integration

Add strict static typing and runtime validation at application boundaries without hiding JavaScript fundamentals.

Major Milestone 2 falls in this sprint.

Self-paced Mon–Fri	– Primitive and object types, functions, unions, literals, narrowing [M] – Interfaces, type aliases, generics, utility types, readonly data [M / H] – Strict compiler configuration; avoiding unsafe any [M] – Typing responses and forms; schema validation at the boundary [M] – Typed error normalisation; authentication headers, retry and pagination patterns [M / H]

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 rather than an obstacle
Sunday Applied Engineering	– DSA Live Clinic — Level 2 opens: two pointers and sliding window – Live build: the API explorer rebuilt as a strict TypeScript application with Zod runtime schemas – Compile-time versus runtime — a deliberate failure demonstration – AI-Native Lab, milestone clinic and sprint review
Projects	– Implementation Project 9 — Typed Data Transformation Toolkit. The logic toolkit converted to strict TypeScript with reusable domain data, errors and configuration modelled as types. – Implementation Project 10 — Typed JavaScript API Application — Major Milestone 2. The API explorer rebuilt as a strict TypeScript application with runtime validation and resilient network handling.
AI-Native Lab	– Toolchain: Claude · Cursor · Anthropic API · Zod · Vercel AI SDK – Typed GenAI client. Learners build an LLM client that returns structured output validated by a Zod schema, then write model-failure test cases: malformed JSON, hallucinated fields, truncated responses, refusals. Prompt and model version are treated as configuration, not as code comments.
DSA	DSA Level 2 opens [M]: two pointers, sliding window, binary search. Prefix sums, stack, queue, linked list and sorting patterns [H].
Evidence	Strict compiler result, typed modules, documented type decisions, typed API client, runtime schemas, search/filter/persistence, tests, live deployment, retrospective, structured-output failure test suite.
Sprint exit	Learner can distinguish compile-time types from runtime validation and build a reliable typed API-driven application.
Scope guardrail	Avoid advanced type-level programming. Types should clarify domain behaviour, not demonstrate cleverness.

SPRINT 06 · **Calendar Weeks 7–8**

React Foundations with TypeScript

Build typed component-based applications with predictable one-way data flow and deliberate component boundaries.

Self-paced Mon–Fri	– Vite setup, environment variables, project structure [H] – React rendering, JSX, component composition, typed props and events [M] – Shared UI and feature folders; MUI components in React [H] – useState, derived values, immutable updates, lifting state [M] – Lists and keys, conditional rendering, state reset [M] – Effects for external synchronisation and cleanup [M]
Saturday Implementation Studio	– Live build: typed component gallery with documented props, variants and states – Component API design in-session — what belongs in props, what does not – Keyboard and screen-reader checks on every component built
Sunday Applied Engineering	– DSA Live Clinic — Level 2: binary search and linked-list patterns – Live build: React product browser with search, filters, selection, loading and error handling – State ownership workshop — moving state up and down live, with consequences visible – AI-Native Lab and sprint review

Projects	– Implementation Project 11 — Typed Component Gallery. Accessible reusable components with documented props, variants, states and composition examples. – Implementation Project 12 — React Product Browser. A React application supporting search, filters, selection, loading and error handling through local state and a typed service layer.
AI-Native Lab	– Toolchain: Claude · Cursor Composer · GitHub Copilot · Vercel AI SDK – AI-assisted component generation under review. Learners generate a component with Cursor Composer, then run a structured review pass: prop-type correctness, accessibility, re-render behaviour, dead code. Every accepted line must be explainable in the sprint review.
DSA	DSA Level 2 continues [M / H]: binary search variants, prefix sums, stack, queue, linked list.
Evidence	Typed component API, gallery pages, keyboard checks, README, feature-based structure, clear state ownership, typed API service, component tests, demo, AI review log.
Sprint exit	Learner can design a typed React component tree, manage local state, and use effects only for external synchronisation.
Scope guardrail	No Redux and no server-cache library yet. State stays local until a demonstrated need appears.

SPRINT 07 · **Calendar Week 9**

React Forms, Routing and Application Data Flow

Create routed applications with accessible validated forms and predictable navigation and data states.

Self-paced Mon–Fri	– Controlled forms, React Hook Form, schema validation [M] – Accessible errors, focus management, multi-step flows [M] – Dynamic fields, server validation mapping; Formik and Yup as an alternative [H / E] – Routes, links, parameters, nested layouts, not-found pages [M] – URL search parameters; protected-route UX and intended destinations [M / H] – Lazy routes and error boundaries [H]
Saturday Implementation Studio	– Live build: multi-step onboarding form with draft persistence and accessible error recovery – Focus-management workshop — the difference a screen reader hears – Submission-state design: pending, partial failure, retry, success
Sunday Applied Engineering	– DSA Live Clinic — Level 2 consolidation and timed mixed set – Live build: product browser and validated workflow combined into a routed application – URL as state — filters made shareable and reloadable in-session – AI-Native Lab and sprint review
Projects	– Implementation Project 13 — Validated Multi-Step Workflow. A multi-step onboarding or application form with draft persistence, clear pending states and accessible error recovery. – Implementation Project 14 — Routed Frontend Application. The product browser and validated workflow combined into a routed application with URL-driven filters and mocked authentication states.
AI-Native Lab	– Toolchain: Claude · Cursor · Vercel AI SDK · Playwright – AI interaction UX. Learners build a streaming assistant interface against a mocked model service, then design fallback UX for invalid, unsafe and truncated outputs. First exposure to prompt-injection risk in a user-facing surface.
DSA	DSA Level 2 completes [M]. Checkpoint 2 assessed at the end of Calendar Week 9.

Evidence	Typed form model, runtime schema, keyboard test, submission-state tests, route map, URL-state tests, route-level loading and error behaviour, pull request, retrospective, DSA Checkpoint 2 result.
Sprint exit	Learner can build a multi-page SPA with validated workflows, predictable navigation and explicit route states.
Scope guardrail	Authentication remains mocked. Security enforcement is not claimed until Sprint 13.

SPRINT 08 · **Calendar Week 10**

Frontend State, Testing, Accessibility and Performance

Select state tools by category, test important behaviour, and improve measured quality without premature complexity.

Self-paced Mon–Fri	– Local, derived, form, URL, global client and server state taxonomy [M] – Context boundaries; Redux Toolkit and RTK Query [M / H] – Classic Redux, Thunk and Saga as awareness [A] – Caching, invalidation and mutations [H] – React Testing Library, user-event, router and state tests [M] – API mocking; render profiling, lazy loading, bundle review [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, not afterwards
Sunday Applied Engineering	– DSA Live Clinic — Level 3 opens: recursion and tree traversal – Live build: high-value component and integration tests written against real user behaviour – Profiling session — measure first, optimise second, measure again – AI-Native Lab and sprint review
Projects	– Implementation Project 15 — 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 16 — Tested Accessible Dashboard. A dashboard with high-value component and integration tests, accessibility checks and one measured performance improvement.
AI-Native Lab	– Toolchain: Claude Code · Cursor · Vitest · Playwright · Promptfoo – AI-generated tests, human-verified. Claude Code proposes a test suite; learners delete every test that passes for the wrong reason, then add the cases the model missed. Introduction to golden test sets using Promptfoo.
DSA	DSA Level 3 opens [H]: recursion, trees, BFS and DFS. Divide-and-conquer intuition, BST, heap and priority queue [H].
Evidence	State inventory, architecture decision note, cache behaviour test, test suite, API mocks, accessibility report, profile evidence, production build review, deleted-test justification log.
Sprint exit	Learner can justify state choices, verify user-visible behaviour, and improve frontend quality using evidence.
Scope guardrail	Performance optimisation must follow measurement. Real-time server integration is deferred until the backend exists.

SPRINT 09 · Calendar Weeks 11–12

Frontend Capstone and Placement Launch

Integrate the frontend pathway into a polished portfolio application and prepare credible evidence for placement activity.

Major Milestone 3 falls in this sprint. This is the frontend completion gate.

Self-paced Mon–Fri	– Problem framing, user stories, acceptance criteria [M] – Component architecture, state plan, test plan, risk control [M] – API mocking for realistic services [H] – Release hardening, accessibility and performance review [M] – Deployment; README and case-study writing [H / M] – JavaScript and React explanation practice for interviews [M]
Saturday Implementation Studio	– Scope approval clinic — instructors cut scope live, with reasoning – Capstone build sprint with architecture review at the midpoint – Risk log maintained in-session
Sunday Applied Engineering	– DSA Live Clinic — Level 3: BST, heap and priority queue – Release hardening, deployment and smoke test performed live – Case-study and portfolio workshop; technical-explanation rehearsal in pairs – AI-Native Lab, milestone review and placement track briefing
Projects	– Implementation Project 17 — Frontend Capstone Alpha. The primary user journey of an original typed React application using validated forms, routing, deliberate state ownership, tests and realistic mocked services. – Implementation Project 18 — Deployed Frontend Capstone and Case Study — Major Milestone 3. Release and present the frontend capstone. Formal frontend-completion gate before placement begins in Calendar Week 13.
AI-Native Lab	– Toolchain: Claude · Cursor · Perplexity – Optional frontend AI feature using a mocked or approved service, shipped with stated limitations and human verification. Learners also produce the first version of their AI Engineering narrative for interviews — what they used, what they rejected, how they verified.
DSA	DSA Level 3 continues [H]. Interview revision block: arrays, strings, hash maps and trees under timed conditions.
Evidence	Approved scope, working alpha, test evidence, pull request review, risk log, live deployment, repository, case study, test report, architecture explanation, resume-ready project summary.
Sprint exit	Learner has a deployed frontend capstone, an evidence-based portfolio story, and approval to begin the placement track.
Scope guardrail	No custom backend during this sprint. Use a mock server or approved external API so frontend quality remains the focus.

PHASE II · Backend Engineering & MERN Integration

SPRINT 10 · **Calendar Week 13**

Node.js and Express Fundamentals

Understand the Node runtime and build a small, typed REST service with reliable configuration and error handling.

Placement activity begins this week and runs to graduation.

Self-paced Mon–Fri	– Event-driven architecture and non-blocking I/O [M] – Process lifecycle and signals; active-LTS Node, scripts, dependency hygiene [H] – Environment validation and graceful shutdown [M] – Request lifecycle, middleware order, routes and controllers [M] – HTTP methods and status codes, validation, standardised errors, API testing [M] – Pagination and filtering basics; Fastify as an alternative framework [H / A]
Saturday Implementation Studio	– Live build: typed Node service with validated configuration, structured startup and shutdown – Health output and a small file or data-processing workflow built in-session – Debugging a Node process live — source maps, signals, leaked handles
Sunday Applied Engineering	– DSA Live Clinic — Level 3 consolidation; Checkpoint 3 preparation – Live build: typed Express REST API with validated requests and centralised errors – Route testing written alongside the routes, not after – AI-Native Lab and sprint review
Projects	– Implementation Project 19 — Node Service Health Toolkit. A typed Node service with validated configuration, structured startup and shutdown behaviour, health output and a small data-processing workflow. – Implementation Project 20 — Typed Express REST API. A small in-memory REST API with validated requests, consistent responses, pagination or filtering, centralised errors and route tests.
AI-Native Lab	– Toolchain: Claude Code · MCP · Cursor · Postman – 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.
DSA	DSA Level 3 completes [H]. Checkpoint 3 assessed at the end of Calendar Week 13, aligned with the start of placement activity.
Evidence	Clean scripts, environment schema, health check, shutdown demonstration, API collection, automated route tests, error examples, README, demo, MCP tool schema with denied-action tests, DSA Checkpoint 3 result.
Sprint exit	Learner can build and explain a typed Express API and the Node process that operates it.
Scope guardrail	In-memory data only. Databases, authentication, queues and WebSockets are intentionally deferred.

SPRINT 11 · Calendar Week 14

API Architecture, Documentation and Testing

Refactor a working API into maintainable layers and verify behaviour through documentation and tests.

Self-paced Mon–Fri	– Controllers, services, repositories, dependency boundaries, domain errors [M] – Adapters, configuration ownership, request IDs [H] – Request and response schemas; route integration tests [M] – OpenAPI documentation and verification [H] – Fixtures, deterministic data, basic contract concepts [H]
Saturday Implementation Studio	– Live refactor: framework-coupled logic extracted into testable services – Architecture diagram drawn in-session and defended by the learner – Structured logging and request IDs added live
Sunday Applied Engineering	– DSA Live Clinic — Level 4 opens: graph representation and traversal – Live build: OpenAPI contract published and verified against the running API – Coverage interpretation workshop — what the number does and does not tell you – AI-Native Lab and sprint review
Projects	– Implementation Project 21 — Layered API Refactor. The REST API refactored into clear layers with framework-coupled logic replaced by testable services. – Implementation Project 22 — Documented and Tested API. A documented API with unit and integration tests and a verified OpenAPI contract.
AI-Native Lab	– Toolchain: Claude Code · MCP · Langfuse · CodeRabbit – Bounded agent groundwork. Learners give an agent loop a stopping rule, a retry budget and an audit log, then attempt to make it misbehave. Agent state, tool authorisation and human approval for sensitive actions are all implemented, not described.
DSA	DSA Level 4 opens [H]: graph representation and traversal, backtracking, greedy, intervals. Topological sort and union-find [E].
Evidence	Architecture diagram, unit tests, request IDs, structured logs, trade-off note, OpenAPI file, test suite, coverage interpretation, CI-ready commands, agent audit log with a documented failed attack.
Sprint exit	Learner can organise an Express codebase, document its contract, and test behaviour at appropriate layers.
Scope guardrail	Architecture proportional to the API. No layers without a current responsibility.

SPRINT 12 · Calendar Weeks 15–16

Database Foundations, MongoDB and Data Access

Compare relational and document data models, implement foundational SQL work, then build and verify efficient MongoDB persistence.

Major Milestone 4 falls in this sprint.

Self-paced Mon–Fri	– Relational concepts: tables, keys, relationships, normalisation, joins [H] – PostgreSQL or MySQL with one selected engine; Sequelize or ORM basics [H / E] – MongoDB documents and collections; embedding versus referencing [M] – Access-pattern modelling, ownership and tenancy, schema evolution, seed data [M / H] – Mongoose schemas, models, validation, pagination [M] – Compound indexes, explain output, atomic updates, isolated database tests [M] – Methods and hooks, optimistic concurrency, aggregation pipelines [H]
Saturday Implementation Studio	– Live modelling: the same domain expressed as a relational schema and a document model, side by side – SQL lab in-session — joins and query plans on a real engine – Trade-off debate, learner-argued and instructor-refereed
Sunday Applied Engineering	– DSA Live Clinic — Level 4: backtracking and greedy – Live build: in-memory storage replaced with MongoDB and Mongoose, indexes verified through explain – One aggregation reporting endpoint built and benchmarked live – AI-Native Lab, milestone clinic and sprint review
Projects	– Implementation Project 23 — Relational vs Document Data Prototype. The same domain modelled as a relational schema and a MongoDB document model, with a CRUD and query lab in PostgreSQL or MySQL and a written trade-off analysis. – Implementation Project 24 — Persistent Data API — Major Milestone 4. In-memory storage replaced with MongoDB and Mongoose, with verified indexes, one reporting endpoint and integration tests.
AI-Native Lab	– Toolchain: MongoDB Atlas Vector Search · Anthropic / OpenAI embeddings · LlamaIndex.TS · Chroma · Claude Code – RAG knowledge service. Learners build ingestion and retrieval end to end — chunking, embeddings, metadata filters, vector search, source attribution — and then test retrieval quality rather than assuming it. Update and delete behaviour must be correct, not just insert.
DSA	DSA Level 4 continues [H]: backtracking, greedy, intervals; topological sort and union-find introduced [E].
Evidence	Relational schema and SQL queries, ORM lab notes, MongoDB model, access-pattern worksheet, persistent API, query-plan evidence, seed script, aggregation endpoint, database tests, retrieval-quality report with failure cases.
Sprint exit	Learner can compare relational and document data models and implement tested persistent data API behaviour.
Scope guardrail	Transactions and complex migrations remain awareness topics unless the project has a clear consistency need.

SPRINT 13 · Calendar Week 17

Authentication, Authorization and Application Security

Protect multi-user applications with complete identity workflows, server-enforced authorization and practical abuse controls.

Self-paced Mon–Fri	– Registration, login, logout, hashing, secure token generation [M] – Verification, recovery and password-reset workflows [H] – Enumeration resistance; cookie sessions versus access and refresh tokens; rotation and revocation [H] – Roles, permissions, ownership, default deny, object-level authorization tests [M] – Validation and secrets handling [M] – Organisation and attribute checks, CORS, secure headers, rate limits, dependency review [H]
Saturday Implementation Studio	– Live build: complete account lifecycle with audit events and failure-case tests – Threat-modelling session on the learner's own API – Cookie and token configuration inspected live in the browser
Sunday Applied Engineering	– DSA Live Clinic — Level 4: intervals and union-find – Live build: backend-enforced ownership and role permissions with tenant-safe queries – Authorization matrix produced and then attacked in pairs – AI-Native Lab and sprint review
Projects	– Implementation Project 25 — Secure Account Lifecycle. The selected authentication model with registration, login, logout, recovery or verification, audit events and tests. – Implementation Project 26 — Secure Multi-User API. Backend-enforced ownership or role permissions, tenant-safe queries, abuse controls and security tests added to the persistent API.
AI-Native Lab	– Toolchain: Claude · Snyk · MCP · Promptfoo – AI security. Learners write a threat model for their AI feature covering direct and indirect prompt injection, data leakage, tool authorisation, tenant isolation and output handling — then build adversarial test cases that must fail safely. Retrieval and tool authorisation are tested per tenant.
DSA	DSA Level 4 continues [H]. Interview revision: BFS/DFS, graphs and backtracking under timed conditions.
Evidence	Threat notes, account-state tests, secure cookie or token settings, audit events, authorization matrix, security tests, dependency review, secrets checklist, adversarial AI test suite with results.
Sprint exit	Learner can distinguish authentication from authorization and prove that protected resources are enforced at the backend boundary.
Scope guardrail	OAuth and OIDC introduced conceptually; provider integration optional unless the project requires it.

SPRINT 14 · Calendar Week 18

Production Backend Workflows, Redis and Real-Time Services

Implement common production workflows beyond CRUD while keeping required scope manageable.

Self-paced Mon–Fri	– Multipart uploads, transactional email, background jobs, external-service integration [E; selected pathway H] – Validation, retries and clear failure behaviour [M] – Streaming, object-storage patterns, dead-letter concepts, provider abstractions [H] – Redis keys and TTLs, cache-aside and invalidation, sessions and idempotency [H / E] – Socket.IO events, rooms, authentication, reconnection, presence [H / E]

Saturday Implementation Studio	– Live build: one production workflow of the learner's choosing, with retries and explicit failure behaviour – Provider abstraction written in-session so the vendor can be swapped – Failure drill — the external service is switched off mid-demo
Sunday Applied Engineering	– DSA Live Clinic — Level 4 consolidation; Checkpoint 4 preparation – Live build: Redis-backed caching or an authenticated Socket.IO feature – Metrics and logs inspected under load – AI-Native Lab and sprint review
Projects	– Implementation Project 27 — One Production Workflow. Choose one: validated upload, transactional email, background job or external-service integration, with retries and clear failure behaviour. – Implementation Project 28 — Caching or Real-Time Feature. Choose one required implementation: Redis-backed caching and session behaviour, or an authenticated Socket.IO feature. The other pathway remains a guided lab.
AI-Native Lab	– Toolchain: BullMQ + Redis · MCP · n8n · Ollama · Claude Code – Agentic workflow orchestration, plus one exposure prototype. Learners run an agent as a background job with queue, retry and dead-letter handling — then choose one of: simulated embodied or robotic control loop, or an edge / autonomous architecture prototype with a fail-safe state and human override.
DSA	DSA Level 4 completes [H]. Checkpoint 4 assessed at the end of Calendar Week 18.
Evidence	Working workflow, provider abstraction, failure test, operational note, automated tests, architecture note, metrics or logs, agent job with dead-letter evidence, exposure prototype with documented safety boundaries, DSA Checkpoint 4 result.
Sprint exit	Learner can deliver one production workflow and one stateful infrastructure feature with explicit failure handling.
Scope guardrail	The choose-one rule is mandatory. Learners are not required to implement uploads, email, queues, Redis and WebSockets in the same sprint.

SPRINT 15 · **Calendar Weeks 19–20**

Full-Stack MERN Integration and Quality

Connect the completed frontend and backend into an authenticated, tested, end-to-end MERN application.

Major Milestone 5 falls in this sprint.

Self-paced Mon–Fri	– Environment boundaries, API clients, authentication state [M] – CORS and cookie or token behaviour, validation mapping, failure-state UX [M] – Optimistic feedback; shared schemas and generated types [H] – Playwright or Cypress high-value flows; service and database test strategy [M] – Deterministic fixtures, regression triage, release notes [H] – Cloud fundamentals: regions, compute, storage, networking, managed services, secrets, cost [H]
Saturday Implementation Studio	– Live integration: one complete frontend journey wired to the secure API and database – Contract mismatch debugging in the browser network panel, live – Failure-state UX built for every network condition demonstrated

Sunday Applied Engineering	– DSA Live Clinic — Level 5 opens: dynamic programming fundamentals – Live build: end-to-end test suite for the primary journey – Beta release cut in-session with release notes and documented limitations – AI-Native Lab, milestone clinic and sprint review
Projects	– Implementation Project 29 — Integrated User Journey. One complete frontend journey connected to the secure API and database, including authentication, authorization, validation and network failure handling. – Implementation Project 30 — Full-Stack MERN Beta — Major Milestone 5. A complete beta release with a small set of reliable end-to-end tests and documented limitations.
AI-Native Lab	– Toolchain: Vercel AI SDK · MongoDB Atlas Vector Search · MCP · Langfuse · Promptfoo – One production-style AI journey integrated into the MERN beta — retrieval plus tool calling plus application workflow — with a golden test set, latency and cost budgets, fallback behaviour and observability. Failure modes must be demonstrated, not asserted.
DSA	DSA Level 5 opens [H / E]: dynamic programming. Mixed patterns and timed problems [M].
Evidence	End-to-end journey, contract note, integration tests, browser network evidence, deployed or demo-ready MERN beta, E2E tests, release notes, retrospective, placement-ready case study, AI evaluation report with failure modes.
Sprint exit	Learner can demonstrate a secure full-stack user journey and explain contracts, data flow, tests and failure states across the stack.
Scope guardrail	One excellent primary journey and one secondary journey beat a large incomplete feature list.

PHASE III · Production Delivery, Capstone & Career Acceleration

SPRINT 16 · **Calendar Week 21**

Containers, CI/CD, Cloud Deployment and Capstone Discovery

Package and release a MERN system through a repeatable pipeline while defining a realistic capstone plan.

Self-paced Mon–Fri	– Docker images and containers, Docker Compose, health checks [M] – Layers, registries, volumes, networks, multi-stage builds, non-root users [H] – Pipeline stages, lint and type checks, tests, builds, secrets, deployment verification, rollback [M] – Pipeline caching, managed services, HTTPS, cost awareness; image scanning [H / E] – Hands-on deployment in one supported cloud platform [H]
Saturday Implementation Studio	– Live build: the full stack containerised — frontend, API, MongoDB, optional Redis or worker – Local failure drill — kill a container, watch the system respond – Capstone problem shortlisting with instructor challenge
Sunday Applied Engineering	– DSA Live Clinic — Level 5: dynamic programming patterns – Live build: automated pipeline to a cloud release, including smoke test and rollback path – Capstone architecture, backlog, threat model, test plan and scope boundary reviewed and approved – AI-Native Lab and sprint review
Projects	– Implementation Project 31 — Dockerized MERN Stack. The full system containerised and running locally with health checks, plus capstone discovery in parallel. – Implementation Project 32 — Automated Cloud Release and Approved Capstone Plan. The MERN beta deployed through an automated pipeline, with an approved capstone architecture, backlog, threat model, test plan, operational plan and scope boundary.
AI-Native Lab	– Toolchain: Ollama · LM Studio · Hugging Face · Docker · Claude – AI deployment and compute fundamentals. Learners compare CPU, GPU, NPU and TPU roles for training versus inference, run a quantized model locally, and produce a compute selection worksheet plus a cloud-versus-edge architecture comparison for their capstone AI feature.
DSA	DSA Level 5 continues [H / E]: dynamic programming and mixed timed sets.
Evidence	Dockerfiles, Compose file, startup documentation, health checks, failure drill, cloud release, CI/CD pipeline, smoke test, rollback path, approved capstone plan, compute selection worksheet.
Sprint exit	Learner can reproduce a containerised local system, release it through a pipeline, and begin a scoped capstone with an approved plan.
Scope guardrail	Managed services and one supported deployment pattern. Advanced orchestration platforms are outside required scope.

SPRINT 17 · Calendar Week 22

Capstone Vertical Slice, Observability and Responsible AI

Deliver the first capstone release with operational visibility and an auditable, supervised AI-assisted workflow.

Major Milestone 6 falls in this sprint.

Self-paced Mon–Fri	– Vertical-slice delivery; structured logs, request IDs, health checks [M] – Backups and restoration testing; metrics, readiness and liveness concepts [M / H] – Incident notes and capacity basics [H / E] – Responsible AI-assisted engineering: prompting with constraints, debugging from evidence, generated-test verification, disclosure, stopping rules [M] – Repository-aware assistants, context preparation, least privilege [H] – Privacy and licensing concerns in AI-assisted development [M]
Saturday Implementation Studio	– Live build: one end-to-end capstone journey across React, Express, MongoDB, authentication and one production workflow – Observability wired in-session — logs, request IDs, health checks – Backup taken and restored live
Sunday Applied Engineering	– DSA Live Clinic — Level 5: mixed patterns under interview conditions – Live build: evaluation harness, prompt and version tracking, retrieval metrics, cost and latency monitoring – AI audit review — rejected suggestions defended in front of peers – Milestone 6 review and sprint retrospective
Projects	– Implementation Project 33 — Capstone Vertical Slice. One end-to-end capstone journey implemented across the full stack with an approved production workflow. – Implementation Project 34 — Operational Capstone Beta and AI Audit — Major Milestone 6. The vertical slice hardened, monitored and exercised through a restoration or rollback drill, with all AI-assisted changes documented and verified.
AI-Native Lab	– Toolchain: Langfuse / LangSmith · Promptfoo · Claude Code · MCP · CodeRabbit – AI Engineering at capstone level. Learners ship an evaluation harness, telemetry, prompt-version tracking, retrieval metrics, model and tool error handling, a tool permission matrix and a rollback or fallback exercise. The AI change log records what was accepted, what was rejected and why.
DSA	DSA Level 5 continues [M]: mixed patterns and timed problems; first full mock coding interview.
Evidence	Working vertical slice, unit/integration/E2E evidence, logs and health checks, backup plan, operational beta, AI prompt and change log, rejected suggestions, test proof, incident or rollback note, tool permission matrix, retrospective.
Sprint exit	Learner can deliver an observable capstone beta and demonstrate accountable use of AI as a supervised engineering assistant.
Scope guardrail	AI-generated code is treated as untrusted until reviewed, tested, understood and disclosed.

SPRINT 18 · Calendar Weeks 23–24

Capstone Completion and Career Accelerator

Harden, release, document and present a production-style MERN application while preparing credible employment and client evidence.

Major Milestone 7 falls in this sprint. Programme completion.

Self-paced Mon–Fri	– Priority journey completion, authorization checks, validation and errors [M] – Security checklist, test stabilisation, release readiness [M] – Index verification, cache rules, retries, resource cleanup, dependency review, go/no-go decisions [H] – Smoke tests, architecture documentation, case study, live demonstration, technical defence [M] – Logs and alerts, backup and rollback exercise, OpenAPI documentation, runbook [H] – Interview stories, handover and client discovery [H / E]
Saturday Implementation Studio	– Release checklist executed live; defect triage and go/no-go decision made as a group – Highest-risk gaps closed with instructor pairing – Test suite stabilised — flaky tests fixed or deleted with justification
Sunday Applied Engineering	– DSA Live Clinic — final mock coding interviews with written feedback – Capstone showcase: live demonstration and technical defence before a panel – AI architecture defence — safety boundaries, human-override story, compute trade-offs – Final portfolio, placement plan and programme close
Projects	– Implementation Project 35 — Capstone Release Candidate. The highest-risk gaps closed, a release checklist run, the test suite stabilised, and a release candidate produced with documented defects and limitations. – Implementation Project 36 — Industry Capstone and Career Presentation — Major Milestone 7. The production-style capstone released and defended: architecture, trade-offs, tests, operations, limitations and AI use.
AI-Native Lab	– Toolchain: Claude · Claude Code · Cursor · full toolchain – Responsible AI-powered software engineering case study. Learners present the final AI architecture section, demonstrate safety boundaries and human override, explain compute and edge trade-offs, and defend a documented record of where AI helped, where it failed and how every output was verified.
DSA	DSA Level 5 completes [M]. Checkpoint 5 assessed, plus two panel mock coding interviews with written feedback.
Evidence	Release checklist, defect triage, security review, query and performance evidence, stable CI, go/no-go note, live application, repository, documentation, operational evidence, portfolio case study, presentation, final interview rehearsal, DSA Checkpoint 5 result.
Sprint exit	Learner has a production-style capstone, a complete portfolio narrative, and evidence for junior frontend or full-stack opportunities.
Scope guardrail	Career claims must match demonstrated evidence. The outcome is junior readiness, not guaranteed employment or production mastery.

10. The AI Engineering Track

The AI track is additive. Every original sprint, project, DSA expectation, milestone, placement activity and assessment standard remains in place; AI capability is threaded through the same programme using focused labs, integrations and capstone extensions rather than replacing engineering fundamentals.

Depth and integration by area

AI area	Depth	Required learner capability	Integration point
Generative AI	H / M in capstone	Prompt and constrain models, validate outputs, manage context, and integrate an AI feature with explicit quality checks.	Sprints 05, 08, 15, 17–18
Agentic AI	H	Build a bounded multi-step agent with tool use, state, stopping rules, retries and human approval for sensitive actions.	Sprints 11, 14, 17–18
AI Engineering	H / M in capstone	Evaluate model behaviour, test AI features, observe latency, cost and errors, design fallbacks, and document production trade-offs.	Sprints 11, 15–18
RAG and vector databases	H	Create an ingestion and retrieval pipeline with embeddings, metadata, vector search, source attribution and retrieval evaluation.	Sprints 12, 15, 17
MCP and tool calling	H	Define tools with schemas, permissions, validation, audit logs and safe execution boundaries; connect an AI workflow to approved tools.	Sprints 04, 10–11, 14, 17
AI-powered software engineering	H / M	Use AI for development while reviewing, testing, verifying, disclosing and rejecting unsafe or unsupported suggestions.	All sprints; assessed at M in 17–18
Edge AI	E / H prototype	Design or simulate local inference under compute, power, privacy, offline and synchronisation constraints.	Sprints 14, 16–17
AI compute and semiconductors	E / H lab	Compare CPU, GPU, NPU, TPU and accelerator trade-offs for training and inference, memory, latency, throughput, quantization and cost.	Sprints 16–17
Robotics and embodied AI	E / H prototype	Explain perception-action loops and implement a simulation or API prototype with safety boundaries and human override.	Sprints 14, 16, 18
Autonomous systems	E / H prototype	Model perception-planning-action-control loops with telemetry, operational boundaries, fail-safe states and human override.	Sprints 14, 16, 18

Required AI portfolio evidence

- One typed Generative AI feature with structured output validation, explicit error and fallback behaviour, and automated tests.
- One RAG implementation using embeddings and vector search, with source attribution, metadata-aware retrieval and retrieval-quality evidence including failure cases.

- One bounded agent or tool-calling workflow with typed schemas, authorization, audit evidence, stopping rules and human approval where appropriate.
- One AI engineering evaluation artifact covering task quality, failure cases, latency, cost, privacy, security and fallback behaviour.
- One compute and edge architecture analysis comparing cloud inference with at least one edge or accelerator-constrained scenario.
- One exposure prototype covering robotics, embodied AI or autonomous systems, including operational boundaries, telemetry, fail-safe behaviour and human override.
- A responsible AI engineering record showing where AI was used, what was rejected, how outputs were verified, and what evidence supports acceptance.

AI scope and safety guardrails

- AI features remain subordinate to core MERN objectives. They extend the applications; they do not replace frontend, backend, database, security, testing, deployment, DSA or career work.
- Model responses, generated code, retrieval results and tool actions require verification proportional to their impact.
- Tool-calling and agent workflows use least privilege, explicit schemas, server-side authorization, input validation, bounded retries, audit logs and human approval for consequential actions.
- RAG systems preserve source attribution and tenant boundaries. Retrieval quality is tested, not assumed.
- Robotics, edge, autonomous-system and semiconductor content is taught through safe software simulation, architecture labs or conceptual exercise unless supervised equipment is explicitly available.

11. 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	Placement value
1	02	Reusable UI System and Marketing Site	Responsive design, accessibility, reusable styling, deployment	Early visual portfolio evidence
2	05	Typed JavaScript API Application	Strict TypeScript, runtime validation, network resilience, tests	JavaScript and TypeScript evidence
3	09	Deployed Frontend Capstone and Case Study	React architecture, forms, routing, state, testing, accessibility, performance	Frontend placement completion gate
4	12	Persistent Data API	Express layers, OpenAPI, data modelling, indexes, integration tests	Backend engineering evidence
5	15	Full-Stack MERN Beta	Secure integrated journey, database, tests, release notes	Full-stack application evidence
6	17	Operational Capstone Beta and AI Audit	Vertical slice, observability, backup and rollback, supervised AI record	Capstone preview for employers
7	18	Industry Capstone and Career Presentation	Production release, documentation, operations, case study, technical defence	Graduate showcase and final portfolio

12. Placement Track

Placement begins in Calendar Week 13, immediately after the frontend capstone gate. Learners apply for frontend internships, junior frontend roles, apprenticeships and project work while backend capability develops. The track is embedded in the timetable rather than added on top of it, consuming roughly three hours per week.

Period	Technical position	Placement activity	Required evidence
Weeks 13–14	Frontend portfolio complete; backend begins	Resume and profile alignment, frontend applications, JavaScript, React and DSA interview practice	Approved resume, portfolio links, application tracker, one mock interview
Weeks 15–17	Database and secure API skills added	Recruiter conversations, take-home practice, accessibility and frontend system questions	Two application reviews, one timed task, one technical story bank
Weeks 18–20	Production workflows and integrated MERN application	Expansion to junior full-stack roles; Node, API, MongoDB and security interviews	Full-stack resume version, MERN beta demo, second mock interview
Weeks 21–22	Deployment and capstone	Employer events, networking, system-	Capstone pitch, architecture

Period	Technical position	Placement activity	Required evidence
	vertical slice	design fundamentals, capstone preview	explanation, interview feedback actions
Weeks 23–24	Production-style capstone complete	Showcase, final interview rehearsal, application follow-up, client-readiness practice	Live presentation, case study, final profile and placement plan

13. 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.

Assessment component	Weight	Evidence
Implementation projects	25%	Thirty-six scoped builds showing continuous application, debugging and improvement across the sprints.
Major milestone projects	25%	Seven deeper submissions assessed for functionality, code quality, accessibility, security, testing and delivery.
Industry capstone and presentation	20%	Architecture, implementation, operational evidence, case study, live presentation and professional defence of decisions.
DSA checkpoints and mock interviews	15%	Five level checkpoints, weekly problem submissions with complexity justification, and three panel mock interviews.
Live studio participation and sprint reviews	10%	Eighteen sprint demonstrations, pull requests, review responses, README updates and retrospectives, plus weekend readiness credit.
Self-paced completion and weekly quizzes	5%	Platform module completion, weekly quizzes and debugging challenges.

Certification thresholds

Requirement	Threshold
Composite score	70 / 100 minimum
Capstone score	6 / 10 minimum, with no individual capstone dimension below 5 / 10
Live studio attendance	80% minimum across the twenty-four weeks
DSA checkpoints	All five attempted; Level 3 checkpoint passed before placement approval
Milestone submissions	All seven submitted; Milestone 3 passed before placement approval

Minimum completion evidence

- Thirty-six implementation project demonstrations or approved make-up submissions.
- Eighteen sprint review records and seven major milestone submissions. Milestones 6 and 7 may be tagged releases of the same capstone repository.
- At least one deployed frontend, one deployed API, and one fully integrated production-style MERN application.
- Automated test evidence across frontend, backend, database and at least one end-to-end user journey.
- A documented security review, operational runbook, backup and restore evidence, and a rollback procedure for the capstone.
- A frontend placement kit by Calendar Week 13 and a full-stack portfolio update by Calendar Week 20.

- A complete AI Disclosure Log covering the full twenty-four weeks, including rejected suggestions and verification method.
- A capstone presentation explaining trade-offs, limitations, rejected approaches and responsible AI use.
- Demonstrated ability to debug unfamiliar failures without relying solely on generated answers.

14. Who This Programme Is For

A strong fit

- Graduates and early-career engineers who want a junior full-stack role and are prepared to give six months to weekend studios and weekday self-study.
- Career changers with some programming exposure who need structure, live accountability and a portfolio that survives scrutiny.
- Working professionals who cannot attend weekday live sessions but can protect Saturday and Sunday mornings.
- Engineers who already use AI assistants casually and want to work with them under professional discipline rather than by improvisation.
- Anyone who intends to sit technical interviews and therefore needs DSA capability assessed and rehearsed, not merely mentioned.

Not a fit

- Anyone seeking a guaranteed placement. This programme provides evidence, coaching and employer access; it does not sell an outcome.
- Learners who cannot commit ten to fifteen hours per week. The pacing is accelerated and the self-paced layer is a prerequisite for the live sessions, not an optional extra.
- Anyone hoping to skip fundamentals because an assistant can generate code. Every accepted line must be explainable in a sprint review.
- Learners looking for depth in foundation-model training, semiconductor design or robotics hardware. Those topics appear here at exposure depth for systems literacy only.
- Anyone who wants a passive video course. Six live hours a week with camera on and code shared is the core of the delivery model.

Impacteers · The AI-Native Full-Stack Engineer Program · Cohort 01

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