FULL STACK ACADEMY

www.fullstackacademy.in 9015 236 236

with Claude AI

3 Months + 2 Weeks Capstone Project

AI-POWERED WEB DEVELOPER

1. MERN Coding Foundations
2. Claude AI for Code Generation
3. Real-Time Apps + Deploy

MERN CODING FOUNDATIONS (WEEKS 1-6)

1. **Introduction —** What You Will Build (AI-Powered First Impressions)
2. **HTML & CSS —** Structure and Style Core HTML: document structure (<!DOCTYPE html>, <html>, <head>, <body>), headings, paragraphs, links, images, divs, spans, and key attributes (id, class, src, href, alt). Semantic elements (<header>, <nav>, <main>, <footer>) for accessibility and SEO. Forms: inputs, textarea, select, checkbox, radio, submit. Tables for data (not layout).
3. **CSS fundamentals:** external stylesheets, selectors (element, class, id, pseudo-classes), the box model, box-sizing: border-box. Colors, Google Fonts, typography. Layout: Flexbox for alignment, CSS Grid for responsive card layouts. Responsive design: viewport meta tag, media queries, mobile-first. CSS Variables, positioning (relative/absolute/fixed/sticky), and smooth transitions. Mini-project: Hand-coded profile card from scratch.
4. **JavaScript —** Fundamentals to Async Core JS: variables (const/let), data types, string methods, arithmetic, comparisons, conditionals, loops, functions, arrow functions, template literals, and scope. Arrays (.map(), .filter(), .reduce(), .find()), objects, destructuring, spread, optional chaining, and nullish coalescing. Error handling with try/catch, ES6 modules. DOM manipulation: querySelector, classList, addEventListener. Async JS: callbacks, Promises, async/await, and the Fetch API for live data. localStorage for persistence. Mini-projects: Quiz app with score tracking; To-do list with filtering and localStorage.
5. **Node.js, Express & REST APIs:** How servers work; installing Node, npm, nodemon, .gitignore. Express routes (GET, POST, PUT, DELETE), route params and query strings, express.json() middleware, custom request logger, dotenv for secrets. Code organisation with routes/ and controllers/. HTTP status codes. Tested with Postman.
6. **MongoDB, Mongoose & JWT Auth SQL vs NoSQL;** MongoDB Atlas setup. Mongoose schemas, models, and full CRUD (.create(), .find(), .findByIdAndUpdate(), etc.) with filter, sort, and limit. Auth: bcrypt password hashing, JWT sign/verify, signup and login routes, protected route middleware. Mini-project: Notes REST API — full CRUD with auth, per-user data.
7. **React — Modern UI Development:** Why React; Vite setup. JSX, functional components, props, conditional rendering. useState for state and re-renders, controlled inputs, event handlers, list rendering with key. useEffect for data fetching; Axios with loading/error states. Advanced hooks: useReducer, useMemo, useCallback, useRef. Context API for shared auth state; custom hooks (useFetch, useAuth). React Router v6: routes, useParams, useNavigate, protected routes. Forms with Formik + Yup. Tailwind CSS for utility-first styling. Mini-project: Full Notes app — CRUD, auth-protected routes, connected to the Express/MongoDB API.

CLAUDE AI FOR CODE GENERATION (WEEKS 7-8)

Claude AI

React: Advanced Patterns: Higher-Order Components (HOCs) for injecting shared behaviour. Optimistic UI updates: reflect changes instantly, roll back on error. Global state with Context + useReducer for auth and notifications. File uploads: FormData, Axios multipart/form-data, image display. Connecting React to Express: Axios interceptors attach JWT automatically, redirect on 401. Performance: React.memo, React.lazy + Suspense for route splitting, image lazy loading. Accessibility: semantic HTML, alt text, keyboard navigation, colour contrast. Mini-project: Polish the Notes app — loading skeletons, empty states, toast notifications, fully mobile-responsive.

FULL STACK ACADEMY

www.fullstackacademy.in 9015 236 236

with Claude AI

3 Months + 2 Weeks Capstone Project

AI-POWERED WEB DEVELOPER

1. MERN Coding Foundations
2. Claude AI for Code Generation
3. Real-Time Apps + Deploy

CLAUDE AI FOR CODE GENERATION (WEEKS 7-8)

Claude AI

Prompt Engineering for Developers: How Claude works: training, tokens, and context window in plain English. Anatomy of an effective prompt: role, context, task, constraints, and output format — applied to a real task immediately. Positive and negative prompting: specifying what to do and what to avoid. Few-shot prompting: 2–3 examples before the ask for dramatically better output. Step-by-step reasoning applied live to a debugging session. Iterative prompting across turns: draft → refine → tighten. Understanding hallucination: why it happens, which developer tasks carry the most risk, and how to spot plausible-but-wrong code before running it. Practical workflows: generating React components, Express routes, and Mongoose schemas; debugging by pasting error + code; refactoring for readability, JSDoc, or async/await. Deliverable: a personal .md prompt library of reusable templates organised by task.

Claude API Integration — Powering Real AI Features Anthropic API fundamentals: the /v1/messages endpoint, request/response structure, and model selection. Security: ANTHROPIC_API_KEY in .env, always proxied through Node — never exposed to the client. First API call from Node: messages array, system prompt, user message, parsing the response. Streaming with Server-Sent Events: word-by-word output displayed live in React. Multi-turn conversations: accumulating the messages array to maintain context across turns. System prompts for role, persona, and output constraints. Token costs and rate limits: estimating cost per feature, handling 429 errors gracefully. Structured JSON output: prompting for JSON, stripping markdown fences, parsing safely, rendering as UI. Three shipped features — a streaming note summariser, an AI Q&A over note content, and smart topic/action-item suggestions. Mini-project: Full AI assistant inside the Notes app — streaming responses, conversation history, and contextual answers.

REAL-TIME, INTEGRATION & DEPLOYMENT (WEEKS 9-12)

WebSockets & Socket.io: Why WebSockets beat polling; Socket.io's reconnection, rooms, and cleaner API. Server/client setup, emit/on on both sides, broadcasting to all/others/rooms, persisting messages to MongoDB, typing indicators and presence. Capstone: stream Claude responses token-by-token through Socket.io.

Full Integration Project — AI-Powered Task Manager MERN + Claude AI + Socket.io combined: MongoDB schemas with relationships, Express auth with role-based access, paginated endpoints, React with protected routes and Tailwind UI, Claude for AI-generated task content, Socket.io for a live collaborative board. End-to-end error handling and a real Git team workflow. Mid-course portfolio piece and capstone foundation.

Deployment & CI/CD Git/GitHub workflow: environment variables in production dashboards, MongoDB Atlas config, Express on Render, React on Vercel with auto-deploy, CORS for HTTP and WebSockets. Custom domains, GitHub Actions for CI/CD, helmet.js/rate-limiting/sanitization for security, UptimeRobot for monitoring. Deliverable: full integration project live — Vercel + Render + Atlas.