FULL STACK ROADMAP

120 concepts to become a full stack developer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

HTML and CSS

Study order: Build three real pages by hand before touching a framework.

01	**Semantic HTML**	header, main, nav and section instead of div everywhere.
02	**Forms**	Inputs, labels and validation attributes.
03	**Box model**	Content, padding, border and margin.
04	**Flexbox**	Lay out items in a row or column.
05	**CSS Grid**	Two-dimensional page layouts.
06	**Responsive design**	Media queries and mobile-first layouts.
07	**Units**	px, rem, %, vh and when to use each.
08	**Positioning**	relative, absolute, fixed and sticky.
09	**Selectors and specificity**	Why one rule beats another.
10	**CSS variables**	Reusable colours and spacing.
11	**Transitions**	Smooth hover and state changes.
12	**Accessibility basics**	Alt text, contrast and keyboard focus.
13	**Tailwind CSS**	Utility classes most new projects use.
14	**Browser DevTools**	Inspect and fix layout bugs live.
15	**Images and performance**	Right sizes and lazy loading.

Practice

1. Build a responsive navbar that collapses on mobile.
2. Centre a card horizontally and vertically in three different ways.

AREA 02 / 08

JavaScript

Study order: Most interview rounds test plain JavaScript, so learn it before React.

01	**let, const and scope**	Block scope and why var is avoided.
02	**Data types**	Primitives vs objects and references.
03	**Functions and arrow functions**	Ways to write and pass functions.
04	**Array methods**	map, filter, reduce and find.
05	**Objects and destructuring**	Pull values out cleanly.
06	**Spread and rest**	Copy and merge arrays and objects.
07	**Closures**	Functions that remember their outer variables.
08	**this keyword**	What this points to in different calls.
09	**DOM manipulation**	Select, create and update elements.
10	**Events**	Listeners, bubbling and delegation.
11	**Promises**	Handle work that finishes later.
12	**async and await**	Write async code that reads top to bottom.
13	**fetch**	Call APIs from the browser.
14	**Event loop**	Call stack, queues and why setTimeout waits.
15	**ES modules**	import and export across files.

Practice

1. What does this print, and why? A closure inside a loop with var.
2. Build a debounced search box without any library.

AREA 03 / 08

React

Study order: Build one app with routing, forms and API calls; that covers most jobs.

01	**Components**	Small reusable pieces of UI.
02	**JSX**	HTML-like syntax inside JavaScript.
03	**Props**	Pass data from parent to child.
04	**State with useState**	Data that changes and re-renders the UI.
05	**useEffect**	Run code after render, like fetching data.
06	**Lists and keys**	Render arrays and why keys matter.
07	**Forms in React**	Controlled inputs and validation.
08	**Lifting state up**	Share state between sibling components.
09	**Context**	Avoid passing props through every level.
10	**Custom hooks**	Reuse logic across components.
11	**React Router**	Pages and navigation in a single-page app.
12	**Data fetching**	Loading, error and empty states.
13	**useMemo and useCallback**	Skip work that doesn't need to rerun.
14	**State libraries**	Redux Toolkit or Zustand for bigger apps.
15	**Next.js basics**	Server rendering and file-based routes.

Practice

1. Build a todo app with add, delete and filter.
2. Why does this component re-render on every keystroke?

AREA 04 / 08

Node and Express

Study order: Write the backend for your React app yourself.

01	**What Node.js is**	JavaScript running on the server.
02	**npm and package.json**	Install and manage packages.
03	**Express setup**	A server in a few lines.
04	**Routes**	GET, POST, PUT and DELETE handlers.
05	**Request and response**	Params, query, body and status codes.
06	**Middleware**	Code that runs before your route.
07	**Error handling**	One place to catch and report errors.
08	**Environment variables**	Keep secrets out of code.
09	**File uploads**	Accept images and documents.
10	**Validation**	Reject bad input before it hits the database.
11	**Logging**	Know what happened when something breaks.
12	**Async errors**	Catch failures in async route handlers.
13	**Folder structure**	Routes, controllers and services.
14	**CORS**	Let your frontend call your backend.
15	**Testing APIs**	Postman and automated tests with Jest.

Practice

1. Build a REST API for notes with all four CRUD routes.
2. Your API is slow on one route. How do you find out why?

AREA 05 / 08

Databases

Study order: Learn one SQL and one NoSQL database; jobs ask for both.

01	**SQL vs NoSQL**	Tables vs documents and when to pick each.
02	**PostgreSQL basics**	Create tables and insert rows.
03	**SELECT and joins**	Query data across tables.
04	**Primary and foreign keys**	Link tables safely.
05	**Indexes**	Make lookups fast.
06	**Normalisation**	Avoid duplicate data.
07	**Transactions**	All-or-nothing writes.
08	**MongoDB basics**	Collections and documents.
09	**Mongoose**	Schemas and models for MongoDB in Node.
10	**ORMs**	Prisma or Sequelize to query from code.
11	**Migrations**	Change the schema safely over time.
12	**Pagination**	Return results page by page.
13	**N+1 queries**	The classic slow-API mistake.
14	**Caching with Redis**	Serve repeated reads from memory.
15	**Backups**	Never lose user data.

Practice

1. Design tables for a food delivery app.
2. A page loads 100 posts and makes 101 queries. Fix it.

AREA 06 / 08

APIs and auth

Study order: Add login to your app and understand every step of it.

01	**REST principles**	Resources, methods and status codes.
02	**JSON**	The format almost every API uses.
03	**Password hashing**	bcrypt; never store plain passwords.
04	**Sessions vs tokens**	Two ways to remember a logged-in user.
05	**JWT**	Signed tokens and what's inside them.
06	**Cookies**	httpOnly and secure flags.
07	**OAuth login**	Sign in with Google.
08	**Authorization**	Roles and who can do what.
09	**Rate limiting**	Stop abuse of login and APIs.
10	**Input sanitising**	Block injection attacks.
11	**HTTPS**	Encrypt traffic between browser and server.
12	**API versioning**	Change APIs without breaking apps.
13	**WebSockets**	Real-time updates like chat.
14	**GraphQL basics**	Ask for exactly the fields you need.
15	**API docs**	Swagger so others can use your API.

Practice

1. Walk through what happens when a user logs in with a JWT.
2. How would you stop someone guessing passwords on your login API?

AREA 07 / 08

Git and deployment

Study order: Ship every project to a live URL; recruiters click links.

01	**Git basics**	init, add, commit and log.
02	**Branches**	Work on features without breaking main.
03	**Merge and rebase**	Two ways to combine branches.
04	**Merge conflicts**	Fix them calmly.
05	**GitHub pull requests**	How teams review code.
06	**.gitignore**	Keep secrets and node_modules out.
07	**Linux commands**	Move around a server terminal.
08	**Environment config**	Different settings for dev and prod.
09	**Frontend hosting**	Vercel or Netlify in minutes.
10	**Backend hosting**	Render, Railway or a cloud VM.
11	**Domains and DNS**	Point your own domain to the app.
12	**Docker basics**	Package the app so it runs anywhere.
13	**CI with GitHub Actions**	Run tests on every push.
14	**Monitoring**	Know when your live app breaks.
15	**Build optimisation**	Smaller bundles, faster loads.

Practice

1. You pushed a bug to main. How do you undo it safely?
2. Deploy a React app and a Node API so they talk to each other.

AREA 08 / 08

Projects and interviews

Study order: Three solid projects beat ten tutorial clones.

01	**Project 1: CRUD app**	Login, forms and a database.
02	**Project 2: real-time app**	Chat or live updates with WebSockets.
03	**Project 3: clone with a twist**	A known app plus one new feature.
04	**README that sells**	Screenshots, live link and what you built.
05	**Portfolio site**	All projects in one place.
06	**DSA basics**	Arrays, strings, hashing and two pointers.
07	**JavaScript output questions**	The tricky snippets interviewers love.
08	**Machine coding round**	Build a small UI feature in an hour.
09	**Low-level design**	Classes and structure for a small system.
10	**System design basics**	Load balancers, caching and databases.
11	**Explaining your project**	Problem, choices and what you'd change.
12	**Resume**	One page, impact first.
13	**LinkedIn and referrals**	Most interviews come through people.
14	**Mock interviews**	Practise out loud with a friend.
15	**Using AI tools**	Copilot speeds you up; you still own the code.

Practice

1. Build an autocomplete search box in 60 minutes.
2. Explain your best project in two minutes.