UnCodemy

Full Stack Developer Roadmap PDF

A full stack developer roadmap moves from HTML, CSS and JavaScript to a front-end framework such as React, then a back end with Node and Express, a database, authentication, APIs, Git, testing and deployment, finished with two or three full projects. This page lays out the stages, what to build in each and how to check progress.

Key takeaways

- Learn the web basics first: HTML, CSS and JavaScript before any framework.
- Add one front-end framework and one back-end stack, and learn them well.
- Understand databases, APIs, authentication and Git early.
- Deploy every project, because deployment teaches real problems.
- Two or three complete projects matter more than many tutorials.

What is full stack developer roadmap?

A full stack developer builds both the user-facing side of a web application (front end) and the server, database and APIs behind it (back end), and can deploy and maintain the whole system.

What does a full stack developer do?

A full stack developer works across the layers of a web application: the interface people see, the server logic that handles requests, the database that stores data and the tools that deploy it. The role differs by company. Some teams expect depth in one layer and working knowledge of the rest.

What are the stages of the roadmap?

Stage 1: web basics. HTML structure, forms, CSS layout with flexbox and grid, responsive design and accessibility. Build a static, responsive portfolio page.

Stage 2: JavaScript. Variables, functions, arrays and objects, the DOM, events, asynchronous code, promises and fetch. Build a to-do app and a small API-driven weather page.

Stage 3: tools. Git and GitHub, the terminal, package managers and a code editor. Practise branches and pull requests on your own projects.

Stage 4: front-end framework. React components, props, state, hooks, forms, routing and data fetching. Build a multi-page app that uses an API.

Stage 5: back end. Node.js, Express, routing, middleware, REST design, validation and error handling. Build a CRUD API.

Stage 6: databases. SQL basics with a relational database and, optionally, a document database. Learn schema design, joins and indexes. Connect your API to a database.

Stage 7: authentication and security. Password hashing, sessions or tokens, authorisation, input validation and common attacks such as injection and cross-site scripting.

Stage 8: testing and quality. Unit tests for functions, component tests and API tests, plus linting and formatting.

Stage 9: deployment. Environment variables, building for production, hosting a front end and a back end, a managed database, a domain and basic logging.

Stage 10: projects and interviews. Two or three full applications with documentation, and practice with data structures, JavaScript questions and system design basics.

What projects fit each stage?

Project	Skills shown
Portfolio site	HTML, CSS, responsive design
To-do app	JavaScript, DOM, local storage
Movie search	Fetch API, async, error states
Blog with login	React, Node, database, auth
Store with cart	State management, API design, payments in test mode
Chat or dashboard	Real-time updates, charts, deployment

How should I choose a stack?

Pick one popular combination and stay with it long enough to build things. A common choice is React, Node, Express and a SQL or document database. The ideas transfer to other stacks, so avoid switching tools often.

How long does the roadmap take?

It varies a great deal with study time and background. A learner studying a couple of hours daily may need many months to cover all stages. Focus on finishing projects rather than hitting a deadline.

How do I know a stage is complete?

You can build the stage's project without copying a tutorial, explain your code and fix a bug in it. If not, repeat the project with a new idea.

What mistakes slow people down?

Tutorial hopping, skipping JavaScript fundamentals, ignoring Git, never deploying, copying code without reading it and building only clones of the same app. Break any project into small features and ship each one.

How do I prepare for jobs?

Write a clear readme for each project, keep your commit history tidy, learn data structures and algorithms basics, practise explaining your design and prepare for questions on JavaScript, React and databases. No roadmap can promise a job; consistent building and clear communication improve your chances.

Which supporting skills help most?

Learn to read documentation and error messages, debug with browser developer tools and server logs, write clear commit messages, and explain technical choices to non-technical people. Basic data structures, such as arrays, maps and sets, and simple algorithms such as sorting and searching are asked in many interviews and also make everyday code better.

How should I keep a learning log?

After each study session note what you built, what broke and how you fixed it. Review the log weekly and turn repeated problems into a short checklist. The log doubles as interview material, because you can describe real bugs you solved.

How do I plan a weekly schedule?

Split the week into learning, building and review. Two sessions for new concepts, two for building the current project, one for reading other people's code and one for review and notes. Keep one rest day. Ship something small every week, such as a new feature or a fixed bug, so that progress is visible.

What should my GitHub profile show?

Pin two or three projects with clear readmes, screenshots, setup steps and a live link. Keep commits meaningful and avoid pushing secrets. A tidy profile lets a reviewer understand your skills in minutes.

Frequently asked questions

What should I learn first for full stack development?

Start with HTML, CSS and JavaScript, then Git, a front-end framework such as React and a back end such as Node with a database.

How many projects do I need?

Two or three complete, deployed projects with documentation are usually more valuable than many tutorial copies.

Which stack should I choose?

Choose one popular combination, such as React with Node, Express and a database, and learn it well before exploring others.

Do I need to know databases?

Yes. Understanding schema design, SQL basics and connecting an API to a database is a core part of full stack work.

Does the roadmap promise a job?

No. It is a learning guide. Hiring depends on the market, your portfolio and interviews.

Sources and further reading

- MDN Learn Web Development
- React documentation
- Node.js documentation
- Git documentation

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550