Journey to FullStackWebDeveloper

Hi there! 😊

Becoming a full-stack web developer means being able to build both the frontend (client

side) and backend (server side) of web applications.

Here's a clear roadmap to help you go from beginner to full-stack developer, step-by-step:

🔹 **Phase 1: Web Basics (Weeks 1–4)**

* Goal: Understand how the web works and build simple static websites.
* HTML – Structure of webpages
* CSS – Styling (colors, layout, fonts, responsiveness)
* Basic Git/GitHub – Version control

✅ Tools to learn:

* VS Code
* Chrome DevTools
* Git + GitHub basics

🔹 **Phase 2: Frontend Development (Weeks 5–10)**

* Goal: Create interactive websites and user interfaces.
* JavaScript (JS) – DOM manipulation, events, arrays, functions, ES6
* Responsive Design – Media queries, Flexbox, Grid
* Framework: React.js (most in-demand)

✅ Tools to learn:

* npm (package manager)
* JSX
* React Hooks
* React Router

🔹 **Phase 3: Backend Development (Weeks 11–16)**

* Goal: Build server-side logic and databases.
* Node.js + Express.js – Backend runtime and web framework
* Databases:
* SQL (e.g., PostgreSQL or MySQL) OR
* NoSQL (e.g., MongoDB)

✅ Key concepts:

* REST APIs
* CRUD operations
* Authentication (JWT, OAuth)

**🔹 Phase 4: Full Stack Integration (Weeks 17–20)**

* Goal: Connect the frontend with the backend to make full applications.
* React frontend + Express backend
* API calling with fetch or axios
* Deployment:
* Frontend: Vercel, Netlify
* Backend: Render, Railway
* Full stack: Heroku, DigitalOcean, or VPS

🔹 **Phase 5: Advanced Topics (Weeks 21–24+)**

* Authentication & Authorization
* File uploads
* WebSockets (real-time apps like chat)
* Testing (Jest, Cypress)
* CI/CD (GitHub Actions)
* TypeScript

🔹 Tools and Best Practices

* Git (advanced branching, merging)
* Docker (basics)
* Debugging and performance optimization
* Soft skills: communication, team collaboration

🧠 How to Practice

* Build projects like:
* To-do app
* Blog
* Chat app
* E-commerce site
* Contribute to open-source on GitHub
* Do coding challenges on LeetCode, HackerRank

**⏱ Estimated Time Based on Daily Study:**

Daily Commitment Total Time Needed Skill Level After

2 hours/day ~10–12 months Job-ready FSD

4 hours/day ~5–6 months Junior to Mid-level

Full-time (6–8 hrs) ~3–4 months Hire-ready fast-track

**🧭 Sample Breakdown (for 2 hrs/day)**

Phase Duration

* ✅ Web Basics (HTML/CSS) 3–4 weeks
* ✅ JavaScript + DOM 4 weeks
* ✅ React (Frontend) 4–5 weeks
* ✅ Node.js + Express 4 weeks
* ✅ Database (MongoDB / SQL) 2–3 weeks
* ✅ Full Stack Integration 3–4 weeks
* ✅ Final Projects + Deployment 3–4 weeks
* ✅ Advanced Topics Ongoing after basics

🏁 Fast-Track Tips:

* Build projects from day 1
* Don't memorize – understand
* Practice coding problems daily (20–30 min)
* Learn by building (todo app → blog → e-commerce)
* day-by-day **full-stack web developer roadmap** assuming 2 hours per day. This plan spans ~6 months (180 days) and is structured into phases, incorporating hands-on learning, projects, and reviews.

📅 Phase 1: Web Basics (Days 1–21)

Goal: Learn HTML, CSS, Git, and how the web works.

* Week 1: HTML + Web Basics
  + Day 1: How the web works (HTTP, browsers, client/server)
  + Day 2: Basic HTML structure, tags ( <div> , <p> , <img> )
  + Day 3: Forms and input elements
  + Day 4: Links, tables, lists
  + Day 5: HTML5 Semantic elements
  + Day 6: Build your first webpage
  + Day 7: Practice and review
* Week 2: CSS
  + Day 8: CSS syntax, selectors, colors, units
  + Day 9: Box model, margins, padding, borders
  + Day 10: Flexbox
  + Day 11: CSS Grid
  + Day 12: Responsive design + media queries
  + Day 13: CSS project: Personal portfolio homepage
  + Day 14: Practice and polish the project
* Week 3: Git & Version Control
  + Day 15: Git basics (init, commit, status)
  + Day 16: GitHub: Push/pull, repo creation
  + Day 17: Branching and merging
  + Day 18: Git in VS Code
  + Day 19: Practice: Create a repo and deploy an HTML/CSS site
  + Day 20: Web deployment (Netlify)
  + Day 21: Review week + mini project

**📘 Phase 2: JavaScript (Days 22–42)**

Goal: Master core JavaScript and DOM manipulation.

* Week 4: JS Basics
  + Day 22: Variables ( let , const ), data types
  + Day 23: Operators, conditionals ( if, switch )
  + Day 24: Loops ( for, while ), arrays
  + Day 25: Functions and arrow functions
  + Day 26: Objects and arrays (deep dive)
  + Day 27: Practice problems
  + Day 28: Mini project (calculator or quiz app)
* Week 5: Advanced JS
  + Day 29: DOM introduction (selectors, elements)
  + Day 30: Events (clicks, inputs)
  + Day 31: Manipulating DOM (text, classes, styles)
  + Day 32: Timing functions, setInterval, setTimeout
  + Day 33: Array methods ( map , filter , reduce )
  + Day 34: LocalStorage + JS mini project (todo list)
  + Day 35: Review and polish the project

**⚛️ Phase 3: Frontend with React (Days 43–70)**

Goal: Build dynamic UIs using React.

* Week 6: React Fundamentals
  + Day 43: What is React? JSX basics
  + Day 44: Components and props
  + Day 45: State and useState
  + Day 46: Events in React
  + Day 47: Conditional rendering
  + Day 48: Lists and keys
  + Day 49: Mini project: React counter or weather app
* Week 7: Intermediate React
  + Day 50: useEffect and lifecycle
  + Day 51: Fetching data from APIs
  + Day 52: Forms and controlled components
  + Day 53: React Router basics
  + Day 54: Props drilling vs context
  + Day 55: Context API intro
  + Day 56: Project: Blog frontend or Recipe app
* Week 8: Component Design
  + Day 57: Reusable components
  + Day 58: CSS in JS (styled-components, Tailwind)
  + Day 59: Responsive React apps
  + Day 60: Deployment with Vercel
  + Day 61–63: Finish & polish full React project
  + Week 9: Review & Challenges
  + Day 64–67: Practice and clone a small site (e.g., YouTube UI)
  + Day 68–70: Review React concepts + Q&A

**🖥 Phase 4: Backend with Node.js & Express (Days 71–98)**

Goal: Build backend APIs and connect with the frontend.

* Week 10: Node.js Basics
  + Day 71: What is Node.js? Setup and intro
  + Day 72: Core modules (fs, path)
  + Day 73: npm, installing packages
  + Day 74: Intro to Express.js
  + Day 75: Basic routes and middleware
  + Day 76: Handling POST requests
  + Day 77: Mini API: Notes or tasks backend
* Week 11: Express Deep Dive
  + Day 78: RESTful APIs
  + Day 79: Status codes, error handling
  + Day 80: Route parameters and query strings
  + Day 81: Middleware, logging, error handling
  + Day 82: Postman testing
  + Day 83–84: Build a REST API (e.g., blog or todo backend)

**🗃 Phase 5: Database (Days 99–113)**

Goal: Store and manage data with MongoDB.

* Week 12: MongoDB & Mongoose
  + Day 99: NoSQL vs SQL, intro to MongoDB
  + Day 100: CRUD operations in MongoDB
  + Day 101: Mongoose setup and schemas
  + Day 102: Creating & retrieving documents
  + Day 103: Updating & deleting docs
  + Day 104: Relationships & population
  + Day 105: Build: Blog database with MongoDB
* Week 13: API + DB Project
  + Day 106–110: Full REST API with DB
  + Day 111: Authentication (JWT basics)
  + Day 112: Authorization (protect routes)
  + Day 113: Review backend

**🌐 Phase 6: Full Stack Integration (Days 114–140)**

Goal: Connect the frontend and backend into one app.

* Week 14: Connecting React + Node
  + Day 114: Set up proxy & API calls
  + Day 115: Connect to MongoDB
  + Day 116: Form submissions from React
  + Day 117–120: Full stack project start (e.g., blog, task manager)
* Week 15: Auth + File Uploads
  + Day 121: Login/register with JWT
  + Day 122: Protected routes (frontend + backend)
  + Day 123: File upload (Multer)
  + Day 124–126: Continue the project
* Week 16: Finalize & Deploy
  + Day 127: Environment variables & security
  + Day 128: Deployment (Render, Railway, Vercel)
  + Day 129–130: Test, fix bugs, polish project

**🚀 Phase 7: Advanced Concepts + Final Projects (Days131–180)**

Goal: Deepen knowledge, prepare for jobs.

* Weeks 17–20: Advanced Topics
  + Day 131–134: TypeScript basics
  + Day 135–138: Unit testing (Jest)
  + Day 139–141: CI/CD with GitHub Actions
  + Day 142–145: WebSockets & real-time chat
  + Day 146–150: Docker basics
* Weeks 21–24: Final Projects
  + Day 151–165: Build a full-featured app (e.g., E-commerce or social media clone)
  + Day 166–170: Polish, test, deploy
  + Day 171–175: Prepare resume, GitHub portfolio
  + Day 176–180: Mock interviews, apply to jobs, keep learning!
* beginner-to-intermediate **project** ideas for each phase in your full-stack learning journey. These will help you practice what you learn, build a portfolio, and reinforce your skills.

**🧱 Phase 1: Web Basics (HTML, CSS, Git)**

Goal: Build beautiful, static websites.

🔹 Project Ideas:

1. Personal Portfolio Website Sections: About Me, Projects, Contact Responsive layout with Flexbox or Grid
2. Simple Landing Page: Product or service landing page. Use attractive colors and layout
3. Resume Template: A clean, printable resume page using HTML/CSS

**⚙️ Phase 2: JavaScript**

Goal: Add interactivity to your websites.

🔹 Project Ideas:

1. To-Do List App: Add, edit, and delete tasks. Store data in localStorage
2. Quiz App: Multiple choice questions, Score tracking, and a timer
3. Calculator: Basic math operations, Real-time display updates
4. Weather App (API) Use a public weather API Display weather by city input

**⚛️ Phase 3: React (Frontend)**

Goal: Build modern, component-based web apps.

🔹 Project Ideas:

1. Task Manager React CRUD with useState, Filter tasks by status
2. Blog UI View posts, search, filter, simulate data with useEffect
3. Recipe Finder Use the API to search and display recipes
4. Movie Browser: Fetch and display data from OMDB or TMDB API

.

**🖥 Phase 4: Node.js & Express (Backend)**

Goal: Build APIs and handle server-side logic.

🔹 Project Ideas:

1. Notes API Routes: GET, POST, PUT, DELETE, In-memory or JSON file storage
2. Blog Backend Posts and user endpoints. Use Postman to test
3. User Authentication API: Register, login, and protected routes using JWT

**🗃 Phase 5: Database (MongoDB)**

Goal: Store and retrieve data from a real database.

🔹 Project Ideas:

1. Full Blog Backend: Posts and comments with MongoDB Mongoose schema + CRUD
2. Product Catalog API: Products, categories, price filtering, Practice query parameters
3. Simple REST API with Users, User registration and login, Save hashed passwords using bcrypt

**🌐 Phase 6: Full Stack Integration**

Goal: Connect frontend and backend for real-world apps.

🔹 Project Ideas:

1. Blog Platform React frontend + Express backend + MongoDB Create, edit, delete posts Login, register, view posts by user
2. To-Do App (Full Stack) Add tasks via frontend, save in DB, Mark as complete/delete, deploy to Vercel + Render
3. Contact Manager: Manage contact list, Auth, forms, validation, Search, and filter contacts

**🚀 Phase 7: Final Projects & Advanced**

Goal: Build impressive portfolio-level apps.

🔹 Project Ideas:

1. E-commerce Store (Mini) Products, cart, checkout, Admin dashboard, Auth, protected routes
2. Social Media App Register/login, posts, likes, comments, Real-time chat with WebSocket (bonus)
3. Project Manager (Trello Clone) Boards, tasks, drag-and-drop, Teams, and roles

✅ **360-day Full Stack Developer Daily Roadmap**, designed for 2 hours/day of study and practice.

This version spreads learning more deeply, includes revision, and provides project-building

days in every phase.

**📘 Phase 1: Web Basics (Days 1–60)**

Topics: HTML, CSS, Git, Basic Layout, Deployment

🔹 Days 1–15: HTML

* Days 1–3: HTML tags, structure, text, media
* Days 4–6: Forms, tables, semantic HTML
* Days 7–9: Practice HTML + mini challenge
* Days 10–12: Create portfolio content in HTML
* Days 13–15: Review & mini project

🔹 Days 16–30: CSS

* Days 16–18: Selectors, colors, box model
* Days 19–21: Flexbox
* Days 22–24: Grid layout
* Days 25–27: Media queries & responsiveness
* Days 28–30: Build a static responsive landing page

🔹 Days 31–45: Git & GitHub

* Days 31–33: Git basics (init, add, commit)
* Days 34–36: GitHub (clone, push, pull)
* 16/21
* Days 37–40: Branching, merging, conflicts
* Days 41–45: Deploy a static site with Netlify

🔹 Days 46–60: Project & Revision

* Days 46–55: Build and style a resume + portfolio website
* Days 56–58: Test and deploy on GitHub Pages or Netlify
* Days 59–60: Review HTML, CSS, Git

**⚙️ Phase 2: JavaScript (Days 61–120)**

Topics: Variables, DOM, Events, Functions, Logic, Storage

🔹 Days 61–80: JavaScript Fundamentals

* Variables, types, operators, loops
* Arrays, functions, objects
* Daily coding practice: small exercises

🔹 Days 81–100: DOM + Events

* Selectors, addEventListener, DOM tree
* Modify content, classes, and styles
* Keyboard/mouse event practice
* Small DOM-based apps: counter, text changer

🔹 Days 101–120: Projects

* Days 101–106: Calculator project
* Days 107–112: Quiz app
* Days 113–117: To-do list with localStorage
* Days 118–120: Review JS with mini challenges

**⚛️ Phase 3: React (Days 121–180)**

Topics: Components, Props, State, Hooks, API Fetching, Router

🔹 Days 121–140: React Basics

* Setup, JSX, components
* Props and useState
* Build small apps: counter, toggle button

🔹 Days 141–160: Intermediate React

* useEffect, data fetching
* Controlled components (forms)
* React Router (pages, navigation)

🔹 Days 161–180: Projects

* Days 161–167: Weather App or Recipe App
* Days 168–174: Movie Browser or Blog UI
* Days 175–180: Polish & review React concepts

**🖥 Phase 4: Node.js + Express (Days 181–240)**

Topics: Backend logic, Routing, Middleware, REST API

🔹 Days 181–200: Node.js + Express

* Setup, modules, basic server
* Routes, HTTP methods
* Use Postman for testing

🔹 Days 201–220: REST API

* CRUD operations
* Error handling, status codes
* Mini API project: Notes or Posts

🔹 Days 221–240: Project

* Days 221–230: Build Blog backend (no DB yet)
* Days 231–240: User Auth API with JWT

**🗃 Phase 5: MongoDB + Mongoose (Days 241–270)**

Topics: Schema, CRUD, Relationships, Mongoose

🔹 Days 241–255: MongoDB Basics

* NoSQL concepts
* Shell commands and Compass
* Integrate MongoDB with Express

🔹 Days 256–270: Project

* Product catalog API (filter, price, category)
* Secure API with password hashing
* Build a database-backed blog API

**🌐 Phase 6: Full Stack Projects (Days 271–315)**

Topics: Combine Frontend + Backend

🔹 Days 271–285: Connect Frontend & Backend

* Proxy setup, fetch data
* Forms to API
* Protected routes and JWT login

🔹 Days 286–300: Project

* Full Stack To-do or Blog (React + Express + MongoDB)
* Features: login, create, edit, delete, display

🔹 Days 301–315: Deployment

* Use Vercel for frontend
* Use Render or Railway for backend
* Environment variables, production builds

🚀 Phase 7: Advanced Topics + Capstone (Days 316–360)

Topics: Auth, Real-time, Testing, Docker, Final Project

🔹 Days 316–330: Advanced

* TypeScript basics
* WebSockets + chat
* GitHub Actions (CI/CD)
* Docker fundamentals

🔹 Days 331–355: Final Project (choose one):

* Social Media App
* E-commerce Store
* Trello-style Project Manager

🔹 Days 356–360: Wrap-up

* Polish portfolio
* Push to GitHub
* Resume + apply to jobs

**daily roadmap for the first 30 days** of the 360-day Full Stack Developer journey:

**📅 Days 1–15: HTML Basics**

* **Day 1:** Learn HTML tags and structure
* **Day 2:** Explore headings, paragraphs, lists
* **Day 3:** Add links and images
* **Day 4:** Create forms with inputs and buttons
* **Day 5:** Build tables and use basic formatting
* **Day 6:** Use semantic elements like <article>, <nav>
* **Day 7:** Practice writing clean HTML
* **Day 8:** Mini challenge: Build a simple webpage
* **Day 9:** Design a basic form layout
* **Day 10:** Add multimedia (audio/video) to a page
* **Day 11:** Use HTML entities and comments
* **Day 12:** Review and clean your code
* **Day 13:** Build a personal profile page
* **Day 14:** Test accessibility and responsiveness
* **Day 15:** Wrap-up: HTML summary and notes

**🎨 Days 16–30: CSS Basics**

* **Day 16:** Learn CSS syntax and selectors
* **Day 17:** Work with colors, fonts, and backgrounds
* **Day 18:** Use the box model and layout spacing
* **Day 19:** Style buttons and inputs
* **Day 20:** Use Flexbox for layout
* **Day 21:** Create multi-column layouts with Grid
* **Day 22:** Add transitions and hover effects
* **Day 23:** Practice responsive design with media queries
* **Day 24:** Design a card layout
* **Day 25:** Responsive navigation bar design
* **Day 26:** Polish a form design with CSS
* **Day 27:** Mini project: Responsive landing page
* **Day 28:** Fix common CSS bugs
* **Day 29:** Review CSS properties and layout systems
* **Day 30:** Wrap-up: CSS review quiz or flashcards

**💻 Days 31–45: Git & Deployment**

* **Day 31:** Install Git and set up GitHub
* **Day 32:** Initialize a Git repository
* **Day 33:** Learn staging and committing files
* **Day 34:** Push code to GitHub
* **Day 35:** Pull and clone a remote repository
* **Day 36:** Create and switch branches
* **Day 37:** Merge branches and resolve conflicts
* **Day 38:** Use .gitignore and view history
* **Day 39:** Practice Git workflow with a sample project
* **Day 40:** Write good commit messages
* **Day 41:** Collaborate using GitHub (fork & pull request)
* **Day 42:** Set up GitHub Pages
* **Day 43:** Deploy a project to Netlify
* **Day 44:** Review Git commands with a cheat sheet
* **Day 45:** Wrap-up: GitHub project showcase

**🌐 Days 46–60: Capstone Project (Resume Website)**

* **Day 46:** Plan your resume website layout
* **Day 47:** Set up HTML structure and sections
* **Day 48:** Add content: skills, experience, education
* **Day 49:** Style sections with CSS
* **Day 50:** Make layout responsive
* **Day 51:** Add a navigation menu
* **Day 52:** Include a contact form
* **Day 53:** Test across devices and fix layout bugs
* **Day 54:** Add animations or effects
* **Day 55:** Commit changes daily using Git
* **Day 56:** Push to GitHub and deploy
* **Day 57:** Add project to GitHub Pages or Netlify
* **Day 58:** Write README with project description
* **Day 59:** Share with peers or community
* **Day 60:** Final polish and review

**⚙️ Days 61–80: JavaScript Basics**

* **Day 61:** Introduction to JavaScript and variables
* **Day 62:** Data types and operators
* **Day 63:** Conditional statements: if, else, switch
* **Day 64:** Loops: for, while, do...while
* **Day 65:** Functions and return values
* **Day 66:** Arrow functions and scope
* **Day 67:** Arrays and array methods (push, pop, etc.)
* **Day 68:** Objects and properties
* **Day 69:** Accessing and modifying nested objects
* **Day 70:** Practice problems (math functions, strings)
* **Day 71:** Debugging with console.log()
* **Day 72:** Template literals and string interpolation
* **Day 73:** Small coding tasks: reverse a string, find max/min
* **Day 74:** Algorithms: factorial, Fibonacci
* **Day 75:** Practice quiz on variables, loops, functions
* **Day 76:** Introduction to ES6 features (let, const, etc.)
* **Day 77:** Refactor earlier code with modern JS
* **Day 78:** Mini project: JavaScript calculator (Part 1 – UI setup)
* **Day 79:** JavaScript calculator (Part 2 – logic implementation)
* **Day 80:** Review key JavaScript concepts

**🧩 Days 81–100: DOM & Events**

* **Day 81:** Introduction to the DOM and document tree
* **Day 82:** Selecting elements using getElementById, querySelector
* **Day 83:** Changing HTML content dynamically
* **Day 84:** Modifying CSS styles via JS
* **Day 85:** Handling button click events
* **Day 86:** Form input events and validation
* **Day 87:** Keyboard and mouse event handling
* **Day 88:** Create a dynamic image gallery or slider
* **Day 89:** Toggle light/dark mode with JS
* **Day 90:** Build simple tabs or accordion interface
* **Day 91:** Create to-do UI structure with HTML & JS
* **Day 92:** Add create/delete functionality to to-do app
* **Day 93:** Store tasks using localStorage
* **Day 94:** Add filters (completed/incomplete)
* **Day 95:** Polish UI for to-do list
* **Day 96:** Start quiz app layout (HTML/CSS)
* **Day 97:** Build quiz logic and question display
* **Day 98:** Add scoring and results tracking
* **Day 99:** Style and debug the quiz app
* **Day 100:** Review DOM, events, and localStorage with a challenge

**⚛️ Days 101–120: React Basics**

* **Day 101:** Introduction to React & setting up with Vite or Create React App
* **Day 102:** Understand components and JSX
* **Day 103:** Build your first component and render it
* **Day 104:** Learn props and how to pass data
* **Day 105:** Create multiple components and reuse them
* **Day 106:** Introduction to useState hook
* **Day 107:** Build a counter app with state
* **Day 108:** Conditional rendering with if, &&, ?
* **Day 109:** Handling user input with forms in React
* **Day 110:** Create a color changer or toggle theme app
* **Day 111:** Introduction to useEffect hook
* **Day 112:** Fetch data from a public API
* **Day 113:** Display fetched data in a list
* **Day 114:** Add loading and error states to API call
* **Day 115:** Start mini project: Recipe Search App (setup + API)
* **Day 116:** Build UI for Recipe Search App
* **Day 117:** Add search functionality and data display
* **Day 118:** Style your app and polish UI
* **Day 119:** Final test + small refactor
* **Day 120:** React recap quiz or flashcard-style review

**⚛️ Days 121–150: Intermediate React**

* **Day 121:** Intro to React Router and route setup
* **Day 122:** Create multiple pages (Home, About, Contact)
* **Day 123:** Link pages using Link from React Router
* **Day 124:** Add a 404 page and dynamic routes
* **Day 125:** Use useParams to fetch dynamic route data
* **Day 126:** Build a multi-page app (e.g. Blog layout)
* **Day 127:** Intro to controlled components (form fields with state)
* **Day 128:** Handle form submissions in React
* **Day 129:** Validate form inputs (basic validation)
* **Day 130:** Practice: Registration or Contact Form
* **Day 131:** Intro to React CRUD (Create, Read, Update, Delete)
* **Day 132:** Implement “Create” part: add item to list
* **Day 133:** Implement “Read” part: list all items
* **Day 134:** Implement “Update” part: edit existing item
* **Day 135:** Implement “Delete” part: remove item
* **Day 136:** Style your CRUD app using CSS or Tailwind
* **Day 137:** Add filters/sorters to list (optional)
* **Day 138:** Refactor app using components
* **Day 139:** Mini Project: Task Manager (Part 1 – Setup & Create)
* **Day 140:** Task Manager (Part 2 – Edit, Delete, UI polish)
* **Day 141:** Add responsiveness and final polish
* **Day 142:** Code cleanup and review CRUD logic
* **Day 143:** Review React Router, Forms, and CRUD in one recap
* **Day 144:** Take a React intermediate quiz
* **Day 145:** Build a mini blog UI or portfolio frontend
* **Day 146:** Add dummy data or mock API for posts
* **Day 147:** Add page switching and post preview
* **Day 148:** Polish layout and navigation
* **Day 149:** Final refactor or challenge (e.g., add like/save)
* **Day 150:** Full review + push to GitHub portfolio

**⚛️ Days 151–180: Full React Projects**

**🌦 Project 1: Weather App (Days 151–157)**

* **Day 151:** Setup project with React + create layout
* **Day 152:** Build search input and connect to OpenWeatherMap API
* **Day 153:** Display current weather: temperature, humidity, icon
* **Day 154:** Add loading/error states and cleanup
* **Day 155:** Style with responsive design
* **Day 156:** Add background change or dark mode based on weather
* **Day 157:** Final review, deploy to Netlify, and push to GitHub

**🎥 Project 2: Movie Browser (Days 158–166)**

* **Day 158:** Create React app with movie card layout
* **Day 159:** Integrate TMDB or OMDB API and fetch movie data
* **Day 160:** Display movies as cards in a grid
* **Day 161:** Add search functionality
* **Day 162:** Add movie details popup using modal or separate route
* **Day 163:** Filter by category/genre
* **Day 164:** Polish UI with CSS or Tailwind
* **Day 165:** Deploy the app to Netlify
* **Day 166:** Push to GitHub and write README

**🍝 Project 3: Recipe Finder (Days 167–174)**

* **Day 167:** Setup app and create recipe search input
* **Day 168:** Connect to a public API (like Edamam or Spoonacular)
* **Day 169:** Display recipe list with images
* **Day 170:** Show details: ingredients, instructions
* **Day 171:** Style with grid layout and animations
* **Day 172:** Add dark mode or favorites feature
* **Day 173:** Final polish, deploy, and test
* **Day 174:** Push to GitHub with a detailed README

**🧼 React Recap & Polish (Days 175–180)**

* **Day 175:** Review useState, useEffect, props
* **Day 176:** Review routing, API calls, forms
* **Day 177:** Clean and optimize all project code
* **Day 178:** Create/update your GitHub portfolio README
* **Day 179:** Take a mock test or build your own React quiz
* **Day 180:** Celebrate! React phase complete 🎉

**🖥️ Days 181–210: Node.js + Express.js**

**🔧 Node.js Fundamentals (Days 181–190)**

* **Day 181:** Intro to Node.js – what is it and why use it
* **Day 182:** Setup Node.js environment and package.json
* **Day 183:** Understand require, module.exports, and CommonJS
* **Day 184:** Read and write files using fs module
* **Day 185:** Build a simple CLI tool or file logger
* **Day 186:** Learn npm – install and manage packages
* **Day 187:** Use external packages (nodemon, chalk, uuid, etc.)
* **Day 188:** Explore asynchronous JS in Node (callback, promise)
* **Day 189:** Practice with APIs and http module
* **Day 190:** Mini project: File-based JSON note keeper CLI

**🌐 Express.js Basics (Days 191–200)**

* **Day 191:** Intro to Express – setup and basic route
* **Day 192:** Create multiple routes (GET, POST, PUT, DELETE)
* **Day 193:** Handle route parameters and query strings
* **Day 194:** Use middleware and logging (e.g. morgan)
* **Day 195:** Send JSON responses and status codes
* **Day 196:** Organize routes in separate files
* **Day 197:** Add basic error handling
* **Day 198:** Learn about RESTful API design
* **Day 199:** Test API using Postman or Thunder Client
* **Day 200:** Mini project: Build a Notes API with in-memory storage

**🔒 Intermediate Express & Auth (Days 201–210)**

* **Day 201:** Setup Express app with JSON parsing and middleware
* **Day 202:** Add file-based data persistence (fs or array)
* **Day 203:** Create a user registration route
* **Day 204:** Learn about hashing with bcryptjs
* **Day 205:** Implement user login with password check
* **Day 206:** Generate JWT tokens and protect routes
* **Day 207:** Use middleware to check tokens
* **Day 208:** Build logout and token expiration logic
* **Day 209:** Refactor code into controllers and middleware
* **Day 210:** Finalize and test: User Auth API project

**🗃️ Days 211–240: MongoDB + Mongoose**

**🧱 MongoDB Fundamentals (Days 211–220)**

* **Day 211:** Intro to NoSQL and MongoDB concepts
* **Day 212:** Install MongoDB locally or use MongoDB Atlas
* **Day 213:** Learn basic Mongo shell commands (insertOne, find, etc.)
* **Day 214:** Explore documents, collections, and databases
* **Day 215:** Understand BSON types, embedded documents, and arrays
* **Day 216:** Perform CRUD operations in the shell
* **Day 217:** Learn querying: filtering, sorting, and projections
* **Day 218:** Indexes and performance basics
* **Day 219:** Practice Mongo queries and data models
* **Day 220:** MongoDB Compass: Visual interface practice

**🔗 Mongoose Integration (Days 221–230)**

* **Day 221:** Setup Mongoose in Express project
* **Day 222:** Define schemas and models
* **Day 223:** Create and read documents with Mongoose
* **Day 224:** Use .save() and .find() methods
* **Day 225:** Add validation to schemas
* **Day 226:** Create relationships with ref and populate
* **Day 227:** Update and delete documents
* **Day 228:** Error handling and status codes
* **Day 229:** Refactor controller logic with async/await
* **Day 230:** Mini project: Build a Blog API with Mongoose

**🛍️ Project: Product Catalog API (Days 231–240)**

* **Day 231:** Design product schema (name, price, category, etc.)
* **Day 232:** Create routes: GET, POST, DELETE, PUT
* **Day 233:** Add filtering by category and search by name
* **Day 234:** Sort products by price, rating, etc.
* **Day 235:** Paginate product list (skip & limit)
* **Day 236:** Add user model and associate products with users
* **Day 237:** Secure product routes with JWT auth
* **Day 238:** Refactor folder structure and add .env
* **Day 239:** Test with Postman and fix edge cases
* **Day 240:** Finalize and push to GitHub with documentation

**Day 241 - Day 260: Advanced Backend Concepts with Node.js & Databases**

**Day 241:** Deep dive into Node.js streams — readable, writable, duplex, and transform streams.  
**Day 242:** Build a file upload system using streams in Node.js with Express and Multer.  
**Day 243:** Explore advanced MongoDB features — Aggregation framework and indexes.  
**Day 244:** Practice designing MongoDB schema with relationships (embedded & referenced docs).  
**Day 245:** Study Redis basics — caching concepts and installing Redis locally.  
**Day 246:** Integrate Redis caching in Node.js backend to speed up data fetching.  
**Day 247:** Learn about message queues — introduction to RabbitMQ or Kafka concepts.  
**Day 248:** Implement a simple job queue with Bull (Redis-based) in your Node.js app.  
**Day 249:** Explore backend security best practices — OWASP top 10 for APIs.  
**Day 250:** Implement input validation and sanitization with Joi or express-validator.  
**Day 251:** Add rate limiting and IP blacklisting using middleware like express-rate-limit.  
**Day 252:** Learn about JWT best practices — refresh tokens and token revocation.  
**Day 253:** Implement role-based access control (RBAC) in your Node.js app.  
**Day 254:** Add logging to backend using Winston or Pino with different log levels.  
**Day 255:** Explore monitoring tools — set up PM2 and basic server health checks.  
**Day 256:** Learn about deployment basics — environment variables, build scripts.  
**Day 257:** Setup CI/CD pipeline basics (GitHub Actions, GitLab CI, or CircleCI).  
**Day 258:** Deploy your backend to a cloud provider — Heroku or Render.  
**Day 259:** Set up a managed MongoDB service like MongoDB Atlas and migrate your DB.  
**Day 260:** Build a simple backend project integrating all above concepts: file upload, caching, RBAC, and deployment.

**Day 261 - Day 280: Advanced Frontend Concepts & React Ecosystem**

**Day 261:** Deep dive into React hooks — useMemo, useCallback, useRef and custom hooks.  
**Day 262:** Learn React Context API for global state management.  
**Day 263:** Explore React Router v6 — nested routes, loaders, and navigation guards.  
**Day 264:** Study component libraries: Material-UI or Chakra UI basics and theming.  
**Day 265:** Practice building responsive layouts with CSS Grid & Flexbox in React apps.  
**Day 266:** Learn React Suspense and lazy loading components for code splitting.  
**Day 267:** Integrate animations using Framer Motion or React Spring.  
**Day 268:** Introduction to testing React components with React Testing Library.  
**Day 269:** Write unit tests for hooks and components with Jest.  
**Day 270:** Learn state management with Redux Toolkit (RTK) — slices, selectors, and middleware.  
**Day 271:** Implement async thunks with RTK Query or Redux Saga for data fetching.  
**Day 272:** Explore form management libraries — Formik or React Hook Form advanced use cases.  
**Day 273:** Build a custom React hook for form validation and error handling.  
**Day 274:** Integrate React with GraphQL (Apollo Client basics).  
**Day 275:** Study progressive web apps (PWA) and add service workers to React app.  
**Day 276:** Learn about React Native basics for mobile app development.  
**Day 277:** Build a simple React Native app to practice cross-platform UI.  
**Day 278:** Explore state management in React Native with Context or Redux.  
**Day 279:** Learn how to test React Native components with Detox or Jest.  
**Day 280:** Deploy React apps — netlify, Vercel, or AWS Amplify.

**Day 281 - Day 300: DevOps & Cloud Fundamentals**

**Day 281:** Understand basics of Linux CLI commands and shell scripting.  
**Day 282:** Learn how to use Git branching strategies — Git Flow or trunk-based development.  
**Day 283:** Study Docker basics — containers, images, Dockerfile creation.  
**Day 284:** Containerize your Node.js backend using Docker.  
**Day 285:** Containerize your React frontend app using Docker.  
**Day 286:** Learn Docker Compose to orchestrate multi-container apps.  
**Day 287:** Introduction to Kubernetes concepts — pods, deployments, services.  
**Day 288:** Deploy your Dockerized app on Kubernetes (minikube/local cluster).  
**Day 289:** Learn about cloud providers — AWS, Azure, Google Cloud overview.  
**Day 290:** Set up an AWS free-tier account and learn S3 for static file hosting.  
**Day 291:** Explore AWS Lambda — serverless functions basics.  
**Day 292:** Build and deploy a serverless API with AWS Lambda and API Gateway.  
**Day 293:** Understand Infrastructure as Code (IaC) — basics of Terraform or AWS CloudFormation.  
**Day 294:** Learn monitoring and alerting basics — Prometheus and Grafana intro.  
**Day 295:** Implement logging aggregation with ELK stack basics (ElasticSearch, Logstash, Kibana).  
**Day 296:** Set up automated backups for databases and files.  
**Day 297:** Explore security in cloud — IAM roles and policies best practices.  
**Day 298:** Learn HTTPS and SSL certificate setup basics (Let's Encrypt).  
**Day 299:** Practice deploying your full app stack on AWS (EC2 + RDS + S3 + Route53).  
**Day 300:** Build a mini DevOps pipeline using GitHub Actions for build, test, deploy.

**Day 301 - Day 320: Real World Full Stack Projects**

**Day 301:** Plan your project: Requirements gathering and tech stack choice.  
**Day 302:** Design database schema for the project.  
**Day 303:** Set up the backend project skeleton with Express and MongoDB.  
**Day 304:** Build authentication and user management system (signup, login, JWT).  
**Day 305:** Build RESTful CRUD APIs for main resources.  
**Day 306:** Write tests for your backend API endpoints.  
**Day 307:** Create React frontend skeleton with routing.  
**Day 308:** Build UI for authentication and user profile management.  
**Day 309:** Integrate frontend with backend auth APIs.  
**Day 310:** Build main app UI — list views, forms, modals.  
**Day 311:** Connect frontend to backend APIs with Axios or Fetch.  
**Day 312:** Implement client-side form validation and error handling.  
**Day 313:** Add real-time features with WebSocket or Socket.io.  
**Day 314:** Add file upload and image handling features.  
**Day 315:** Implement notifications system (email or in-app).  
**Day 316:** Add payment gateway integration (Stripe or PayPal sandbox).  
**Day 317:** Optimize app performance and code splitting.  
**Day 318:** Add unit and integration tests for frontend components.  
**Day 319:** Deploy your full app to production (frontend + backend).  
**Day 320:** Write project documentation and README.

**Day 321 - Day 340: Open Source Contribution & Community**

**Day 321:** Find open source projects that interest you on GitHub.  
**Day 322:** Learn how to fork, clone, and submit pull requests (PRs).  
**Day 323:** Start by fixing small issues or bugs in open source projects.  
**Day 324:** Participate in discussions and community forums for projects.  
**Day 325:** Contribute documentation or write examples for open source repos.  
**Day 326:** Learn about Semantic Versioning and Git tags in open source projects.  
**Day 327:** Try contributing a new feature to a project.  
**Day 328:** Attend online meetups or webinars in your tech stack.  
**Day 329:** Write blog posts or tutorials about what you've learned.  
**Day 330:** Create a personal portfolio website showcasing your projects.  
**Day 331:** Learn about SEO basics and improve your portfolio site.  
**Day 332:** Add your open source contributions to your resume and LinkedIn.  
**Day 333:** Practice technical interview questions — algorithms & data structures.  
**Day 334:** Prepare your elevator pitch and talk about your projects confidently.  
**Day 335:** Practice coding challenges on HackerRank, LeetCode, or Codewars.  
**Day 336:** Network with developers on LinkedIn or Twitter.  
**Day 337:** Review your GitHub profile and clean up your repositories.  
**Day 338:** Plan a small community event or study group with peers.  
**Day 339:** Learn about freelancing platforms and how to get your first client.  
**Day 340:** Apply for internships, junior dev roles, or freelance gigs.

**Day 341 - Day 360: Mastery and Polishing**

**Day 341:** Revisit and refactor your earlier projects with new knowledge.  
**Day 342:** Improve UI/UX with accessibility and internationalization (i18n).  
**Day 343:** Learn advanced debugging techniques and tools.  
**Day 344:** Deep dive into GraphQL advanced topics — subscriptions and batching.  
**Day 345:** Explore microservices architecture basics.  
**Day 346:** Experiment with NoSQL vs SQL — build small apps using PostgreSQL.  
**Day 347:** Learn about backend performance tuning and profiling.  
**Day 348:** Study frontend performance best practices — Lighthouse audit.  
**Day 349:** Implement offline support in your app (Service Workers, IndexedDB).  
**Day 350:** Learn basics of AI/ML APIs integration (like OpenAI or TensorFlow.js).  
**Day 351:** Practice pair programming with a fellow developer.  
**Day 352:** Review your entire learning journey and list your strengths.  
**Day 353:** Plan next steps for continuous learning — frameworks, languages.  
**Day 354:** Set up a blog or YouTube channel to teach what you know.  
**Day 355:** Build a community or join mentorship programs.  
**Day 356:** Experiment with new tech trends — Web3, blockchain basics.  
**Day 357:** Practice soft skills — communication, teamwork, time management.  
**Day 358:** Prepare a comprehensive portfolio presentation.  
**Day 359:** Conduct mock interviews with peers or mentors.  
**Day 360:** Celebrate your journey! Plan your career roadmap ahead.

Congratulation