**Web-Dev 2.0 Roadmap (120 Days Challenge)**

**Overview of the 4-Month Roadmap**

1. **Month 1 (Days 1–30):**
   * **Focus:** HTML5, CSS3, Git/GitHub, and Core JavaScript (ES6+)
   * **Goal:** Solid grasp of front-end basics and version control.
2. **Month 2 (Days 31–60):**
   * **Focus:** Advanced JavaScript, React basics, React Hooks, and small React projects
   * **Goal:** Comfortably build modern front-end applications with React.
3. **Month 3 (Days 61–90):**
   * **Focus:** Node.js, Express.js, MongoDB, building RESTful APIs
   * **Goal:** Back-end proficiency with a Node/Express/MongoDB stack.
4. **Month 4 (Days 91–120):**
   * **Focus:** Integration of front-end and back-end, advanced topics, deployment, and final Capstone Project
   * **Goal:** Become a full-stack developer able to create, deploy, and showcase a complete application.

**Month 1: HTML, CSS, Git, and Core JavaScript (Days 1–30)**

**Week 1: HTML Essentials and Introduction to Git/GitHub (Days 1–7)**

**Day 1: HTML Fundamentals**

* Topics to Learn:
  1. Structure of an HTML document (<!DOCTYPE html>, <html>, <head>, <body>)
  2. Common tags (<p>, <h1>–<h6>, <span>, <div>)
  3. Semantic HTML5 tags (<header>, <nav>, <section>, <article>, <footer>)
* Assignment:
  1. Create a simple HTML page with a header, a main content area, and a footer.
* Resources:
  1. MDN HTML Basics

**Day 2: HTML Forms and Media**

* Topics to Learn:
  1. Forms (<form>, <input>, <select>, <textarea>, <button>)
  2. Media embedding (<img>, <video>, <audio>)
  3. Accessibility basics (ARIA labels, form accessibility)
* Assignment:
  1. Build a contact form with name, email, message fields, and a submit button.
* Resources:
  1. MDN: HTML Forms

**Day 3: Git/GitHub Setup**

* Topics to Learn:
  1. Installing Git
  2. Basic Git commands (git init, git add, git commit, git status, git log)
  3. Creating a GitHub repository and pushing your local repo
* Assignment:
  1. Initialize a Git repo for the mini HTML projects and push to a GitHub repository.
* Resources:
  1. Official Git Documentation

**Day 4: Git Branching and Collaboration**

* Topics to Learn:
  1. Branching workflow (git branch, git checkout -b)
  2. Merging branches (git merge)
  3. Resolving merge conflicts
* Assignment:
  1. Create a new branch for a feature, make changes, merge it back to main (or master).
* Resources:
  1. Atlassian Git Tutorial on Branching

**Day 5: Semantic HTML Deep Dive**

* Topics to Learn:
  1. More semantic tags (<figure>, <figcaption>, <aside>)
  2. HTML best practices, SEO basics (metadata, <meta> tags)
* Assignment:
  1. Add semantic tags and meta tags to your HTML pages for improved structure and SEO.

**Day 6: Mini Project #1 – Simple Multi-Page Website**

* Project Outline:
  1. Home page + About page + Contact page
  2. Navigation bar across pages
  3. Integrate simple form on the contact page
  4. Use Git to track your changes
* Deliverable:
  1. Host the mini-project’s code on GitHub.

**Day 7: Review & Reinforcement**

* Tasks:
  1. Review all HTML topics and Git commands from the past 6 days
  2. Clean up any code or Git repositories
  3. Reflect on your learning; note any gaps or confusion
* **Optional**: Explore advanced HTML topics like <template>, <canvas> if time permits

**Week 2: CSS – Styling the Web (Days 8–14)**

**Day 8: CSS Basics**

* Topics to Learn:
  1. Inline vs. Internal vs. External CSS
  2. Selectors, Specificity
  3. The Box Model (margin, border, padding)
* Assignment:
  1. Create a CSS file and link it to your previous HTML mini-project. Experiment with basic styling.
* Resources:
  1. MDN: CSS First Steps

**Day 9: CSS Positioning and Display**

* Topics to Learn:
  1. position property (static, relative, absolute, fixed, sticky)
  2. display property (block, inline-block, inline, flex, grid)
* Assignment:
  1. Practice placing elements in various positions. Understand how z-index works.

**Day 10: Flexbox**

* Topics to Learn:
  1. Container and items (display: flex;, justify-content, align-items, flex-direction)
  2. Real-world layouts with flexbox (navbar, card layouts)
* Assignment:
  1. Build a responsive navbar or card layout using Flexbox.
* Resources:
  1. Flexbox Froggy Game

**Day 11: CSS Grid**

* Topics to Learn:
  1. Basic grid properties (grid-template-columns, grid-template-rows, gap)
  2. Grid vs. Flexbox usage scenarios
* Assignment:
  1. Create a two-dimensional layout (like a gallery) using CSS Grid.
* Resources:
  1. Grid Garden

**Day 12: Responsive Design & Media Queries**

* Topics to Learn:
  1. Mobile-first design strategy
  2. @media queries (breakpoints for different screen sizes)
  3. Responsive units (%, em, rem, vw, vh)
* Assignment:
  1. Make your previous layouts responsive for mobile and tablet devices.

**Day 13: CSS Best Practices & Animations**

* Topics to Learn:
  1. BEM methodology (Block, Element, Modifier) for structuring CSS
  2. Basic CSS animations (@keyframes, transition, transform)
* Assignment:
  1. Add a simple hover animation or transition effect to a button or image.

**Day 14: Mini Project #2 – A Responsive Portfolio Page**

* Project Outline:
  1. Single-page portfolio with sections: “Home”, “About”, “Projects”, “Contact”
  2. Use Flexbox or Grid for layout
  3. Ensure full responsiveness across devices
  4. Animate some elements (e.g., fade-in sections)
* Deliverable:
  1. Deploy the page on Netlify or GitHub Pages

**Week 3: JavaScript Fundamentals (Days 15–21)**

**Day 15: JavaScript Basics**

* Topics to Learn:
  1. Variables (let, const, var)
  2. Data types (string, number, boolean, null, undefined, object)
  3. Operators (+, -, \*, /, %, \*\*, etc.)
* Assignment:
  1. Write a script that performs basic arithmetic operations and logs outputs to console.
* Resources:
  1. MDN: JavaScript Basics

**Day 16: Control Flow**

* Topics to Learn:
  1. Conditionals (if/else, switch)
  2. Loops (for, while, do...while)
* Assignment:
  1. Create a script to process user input in the console using conditionals and loops.

**Day 17: Functions and Scope**

* Topics to Learn:
  1. Function declarations vs. expressions, arrow functions
  2. Global scope vs. local scope
  3. Parameter defaults
* Assignment:
  1. Write a function library that contains reusable utility functions (e.g., random number generator).

**Day 18: Arrays and Objects**

* Topics to Learn:
  1. Array methods (push, pop, slice, splice, map, filter, reduce)
  2. Object literal syntax, nested objects, object methods
* Assignment:
  1. Implement an address book array of objects. Let users add, remove, and search contacts.

**Day 19: DOM Manipulation**

* Topics to Learn:
  1. Selecting elements (document.querySelector(), document.getElementById())
  2. Modifying styles and content
  3. Event handling (click, input, change)
* Assignment:
  1. Create a simple to-do list or counter app that manipulates the DOM based on user interactions.
* Resources:
  1. MDN: DOM Introduction

**Day 20: JSON and LocalStorage**

* Topics to Learn:
  1. JSON syntax and usage
  2. JSON.stringify(), JSON.parse()
  3. Storing data in localStorage or sessionStorage
* Assignment:
  1. Enhance the to-do list by persisting tasks in localStorage.

**Day 21: Mini Project #3 – Interactive Web App**

* Project Outline:
  1. A small “Quiz App” or “Recipe Manager” or “Weather Fetcher” (using a free weather API)
  2. Use DOM manipulation extensively
  3. Store user data in LocalStorage
* Deliverable:
  1. Host the project code on GitHub and share the live link

**Week 4: Advanced JS Concepts and Consolidation (Days 22–30)**

**Day 22: ES6+ Features**

* Topics to Learn:
  1. Destructuring, spread operator, rest parameters
  2. Template literals, shorthand property names
* Assignment:
  1. Refactor a previous project using these modern JS features.
* Resources:
  1. MDN: ES6 In Depth

**Day 23: Asynchronous JavaScript**

* Topics to Learn:
  1. Callbacks
  2. Promises (.then(), .catch())
  3. async/await syntax
* Assignment:
  1. Build a simple app that fetches data from a public API using fetch() or axios.

**Day 24: Error Handling & Debugging**

* Topics to Learn:
  1. try...catch blocks
  2. DevTools debugging (breakpoints, watch variables)
* Assignment:
  1. Debug a piece of broken code and properly handle errors.

**Days 25–26: Mini Project #4 – API-Driven Web App**

* Project Outline:
  1. Example: “Movie Search App” that hits an external movie API
  2. Implement loading states, error states
  3. Use async/await or Promises for requests
* Deliverable:
  1. Deploy or host the project (Netlify/GitHub Pages).

**Day 27: Git/GitHub Deep Dive**

* Topics to Learn:
  1. Pull requests, code reviews
  2. GitHub Issues & Projects for task management
  3. Git tags and releases
* Assignment:
  1. Create a feature branch, open a Pull Request, review and merge it.

**Day 28: Refactoring & Code Organization**

* Topics to Learn:
  1. Module patterns (ES Modules, CommonJS basics)
  2. File/folder structure for web apps
* Assignment:
  1. Split your project code into multiple JS modules.

**Days 29–30: Final Review of Front-End Foundations**

* Tasks:
  1. Revisit all HTML/CSS/JS fundamentals
  2. Clean up portfolio projects and push them to GitHub
  3. Prepare to dive into React next week

**Month 2: React.js & Advanced Front-End (Days 31–60)**

**Week 5: React Basics (Days 31–37)**

**Day 31: Introduction to React**

* Topics to Learn:
  1. Setting up a React app using Create React App (CRA) or Vite
  2. React folder structure
  3. JSX syntax
* Assignment:
  1. Create a basic React app, display “Hello World” with a functional component.
* Resources:
  1. React Official Docs

**Day 32: Components & Props**

* Topics to Learn:
  1. Functional components vs. Class components (focus on functional for modern React)
  2. Props usage, prop types
* Assignment:
  1. Build a small “Profile Card” component that takes in props (name, picture, description).

**Day 33: State and Lifecycle**

* Topics to Learn:
  1. Using useState Hook for state management
  2. Basic lifecycle with hooks (e.g., useEffect)
* Assignment:
  1. Enhance the “Profile Card” with a “Follow” button that toggles follow state.

**Day 34: Event Handling in React**

* Topics to Learn:
  1. onClick, onChange, onSubmit in React
  2. Synthetic event system
* Assignment:
  1. Create a simple form in React that updates state on user input.

**Day 35: Conditional Rendering & Lists**

* Topics to Learn:
  1. Rendering lists with .map()
  2. Handling unique keys (key prop)
  3. Conditional rendering (&&, ?:)
* Assignment:
  1. Display a list of items (e.g., tasks or product data). Conditionally show a message if the list is empty.

**Day 36: Basic React Router**

* Topics to Learn:
  1. React Router setup
  2. Creating multiple routes (<BrowserRouter>, <Routes>, <Route>)
  3. Navigation with <Link> or useNavigate
* Assignment:
  1. Convert your multi-page portfolio to a single-page React app using React Router.
* Resources:
  1. React Router Docs

**Day 37: Mini Project #5 – Simple React App**

* Project Outline:
  1. A small “Product Catalog” or “Task Manager” with at least two routes
  2. Use state to add/remove/update items
  3. Show mastery of basic React, props, state, router
* Deliverable:
  1. Deploy the React app on Netlify or Vercel

**Week 6: Deeper into React (Days 38–44)**

**Day 38: Advanced Hooks**

* Topics to Learn:
  1. useEffect for side effects (API calls, subscriptions)
  2. useContext for global state
  3. useReducer for complex state logic
* Assignment:
  1. Refactor your mini project to use useContext for theme or user auth state.

**Day 39: Forms and Form Libraries**

* Topics to Learn:
  1. Controlled vs. uncontrolled components
  2. Libraries: Formik or React Hook Form
* Assignment:
  1. Build a “Sign Up” form with validation using Formik or React Hook Form.

**Day 40: Styling in React**

* Topics to Learn:
  1. CSS Modules, Styled Components, or Tailwind CSS
  2. Pros and cons of various styling approaches
* Assignment:
  1. Add a consistent UI library or styling approach to your mini projects.

**Day 41: State Management (Redux or Context + Reducer)**

* Topics to Learn:
  1. Redux basics (store, actions, reducers) OR advanced usage of React Context + Reducer pattern
* Assignment:
  1. Implement a global store for your existing React project (e.g., a cart for an e-commerce site).

**Days 42–43: Mini Project #6 – React CRUD App**

* Project Outline:
  1. A “CRUD” (Create, Read, Update, Delete) app, such as a “Contact Manager” or “Blog Posts”
  2. Use advanced hooks, React Router, form validations
  3. Consider using Redux or Context for state
* Deliverable:
  1. Hosted project link plus GitHub repo

**Day 44: Review & React Best Practices**

* Tasks:
  1. Optimize components (memoization, pure components)
  2. Folder structure best practices (e.g., feature-based)
  3. Prepare to transition into the Node.js ecosystem
* Resource:
  1. React Performance Optimization

**Week 7: Moving to the Back-End – Node.js Fundamentals (Days 45–51)**

*(Adjusting day numbers to align with the 120 total. You have 60 days allocated for front-end. We are at Day 44, so continuing…)*

**Day 45: Node.js Introduction**

* Topics to Learn:
  1. Node.js architecture, V8 engine
  2. Setting up Node, running a simple .js file
  3. npm vs. yarn
* Assignment:
  1. Create a simple script in Node that reads a file and logs its content to the console.
* Resources:
  1. Node.js Official Docs

**Day 46: Basic Node Modules and NPM**

* Topics to Learn:
  1. Importing and exporting modules (require, module.exports, or ES Modules)
  2. Common built-in modules (fs, path)
  3. Creating your own modules
* Assignment:
  1. Build a small utility module and import it into a main file.

**Day 47: Express.js Basics**

* Topics to Learn:
  1. Setting up an Express server
  2. Routing (GET, POST, etc.)
  3. Middleware basics
* Assignment:
  1. Create a simple Express server with at least two routes.

**Day 48: Templating Engines & Static Files**

* Topics to Learn:
  1. Serving static files (images, CSS) with Express
  2. Using templating engines (EJS, Pug, or Handlebars) – although in a React-based stack, you might not need these often
* Assignment:
  1. Build a basic server-rendered web page using EJS or Handlebars.

**Days 49–50: Express Router & MVC Structure**

* Topics to Learn:
  1. Splitting routes into separate files
  2. Organizing code into MVC (Model, View, Controller) patterns
* Assignment:
  1. Convert your simple Express app into a more modular structure with separate routes.

**Day 51: Mini Project #7 – RESTful API**

* Project Outline:
  1. Build a small CRUD API for a resource (e.g., “Books” or “Movies”)
  2. Use Postman or a similar tool to test your endpoints
  3. Deploy your simple API on Railway or Heroku (if free tier is available)
* Deliverable:
  1. Live API endpoint and GitHub repo

**Month 3: Full Back-End (Node, Express, MongoDB) & Integration (Days 61–90)**

**Week 8: MongoDB & Mongoose (Days 61–67)**

**Day 61: MongoDB Introduction**

* Topics to Learn:
  1. NoSQL concepts, document-based databases
  2. Installing and running MongoDB locally or using MongoDB Atlas
* Assignment:
  1. Create a test database and insert a few documents.
* Resources:
  1. MongoDB University Free Courses

**Day 62: Mongoose ODM**

* Topics to Learn:
  1. Connecting to MongoDB via Mongoose
  2. Defining schemas and models
* Assignment:
  1. Create a “User” model with Mongoose and perform basic CRUD operations.

**Day 63: Advanced Queries**

* Topics to Learn:
  1. Filtering, projections
  2. Pagination and sorting
* Assignment:
  1. Build a route in Express that fetches paginated data from MongoDB.

**Day 64: Data Validation and Error Handling**

* Topics to Learn:
  1. Schema validation with Mongoose
  2. Handling errors gracefully in Express
* Assignment:
  1. Add validation rules to your “User” or “Book” model, handle invalid data.

**Day 65: Authentication (JWT Basics)**

* Topics to Learn:
  1. JSON Web Tokens (JWT)
  2. Generating and verifying tokens
  3. Protected routes in Express
* Assignment:
  1. Implement a login and signup route that returns a token upon successful authentication.

**Days 66–67: Mini Project #8 – Secure REST API with MongoDB**

* Project Outline:
  1. Build an “Auth-Enabled” CRUD API for a resource of your choice
  2. Users can only read/write their own data (middleware for authentication)
  3. Proper error handling and validation
* Deliverable:
  1. Deployed API with docs, GitHub repo

**Week 9: Integrating Front-End & Back-End (Days 68–74)**

**Day 68: Connecting React Front-End with Express Back-End**

* Topics to Learn:
  1. RESTful API calls from React (using fetch or axios)
  2. CORS issues and how to fix them
* Assignment:
  1. Build a simple React front-end that consumes your secure REST API from the previous mini-project.

**Day 69: Authentication on the Client**

* Topics to Learn:
  1. Storing JWT in localStorage/cookies
  2. Auth context or Redux-based auth flow
* Assignment:
  1. Protect certain pages in React so only authenticated users can access them.

**Day 70: Handling File Uploads**

* Topics to Learn:
  1. Multer or similar middleware in Express
  2. Handling file uploads in React forms
* Assignment:
  1. Create an endpoint to upload profile pictures and display them in a React app.

**Days 71–72: Mini Project #9 – Full-Stack Application**

* Project Outline:
  1. A “Full-Stack Dashboard” or “Task Manager” with user authentication
  2. CRUD functionality on tasks or data, stored in MongoDB
  3. React front-end with protected routes
* Deliverable:
  1. Deployed front-end (Netlify/Vercel) and back-end (Railway/Heroku)

**Day 73: GitHub Project Management**

* Topics to Learn:
  1. Creating Issues, linking PRs to Issues
  2. Using GitHub Projects or Kanban boards
* Assignment:
  1. Manage your mini-project tasks using GitHub’s project board.

**Day 74: Code Review & Optimization**

* Topics to Learn:
  1. Minimizing bundle size (code splitting in React)
  2. Using environment variables for sensitive data
* Assignment:
  1. Implement environment variables for your API keys or database URIs.

**Week 10: Advanced Back-End & Third-Party Integrations (Days 75–81)**

**Day 75: Advanced Express Concepts**

* Topics to Learn:
  1. Rate limiting (e.g., express-rate-limit)
  2. Security best practices (Helmet, sanitize inputs)
* Assignment:
  1. Add security measures to your existing Express app.

**Day 76: Working with Third-Party APIs**

* Topics to Learn:
  1. OAuth flows (basic understanding)
  2. Integrating with a popular API (e.g., Twitter, Stripe, or PayPal)
* Assignment:
  1. Create a route that fetches data from a third-party API or processes payments with Stripe test mode.

**Day 77: Logging and Monitoring**

* Topics to Learn:
  1. Using morgan for request logging
  2. Error logging with services like Sentry (optional)
* Assignment:
  1. Enhance your app to log requests and errors.

**Days 78–79: Mini Project #10 – Third-Party Integration**

* Project Outline:
  1. Extend your previous full-stack app to include a third-party API feature (e.g., payment, social login)
  2. Ensure it’s secure, handle errors properly
* Deliverable:
  1. Deployed updated full-stack app

**Day 80: Testing Basics**

* Topics to Learn:
  1. Unit testing with Jest or Mocha/Chai
  2. Integration tests for Express routes
* Assignment:
  1. Write tests for at least one model and one route in your API.

**Day 81: Review & Prep for Advanced Topics**

* Tasks:
  1. Revisit your full-stack mini projects
  2. Document any existing gaps (testing, security, performance)
* Resource:
  1. Jest Documentation

**Month 4: Final Polish, Advanced Topics, and Capstone Project (Days 82–120)**

**Week 11: Advanced Deployment & Production Readiness (Days 82–88)**

**Day 82: Deployment Options**

* Topics to Learn:
  1. Comparing hosting platforms (AWS, DigitalOcean, Render, Vercel)
  2. Docker basics (optional deep dive)
* Assignment:
  1. Deploy a test app using a new service (e.g., Render or AWS EC2).

**Day 83: CI/CD Pipelines**

* Topics to Learn:
  1. GitHub Actions or GitLab CI
  2. Automating tests and deployments
* Assignment:
  1. Configure a simple pipeline that runs your tests on every push.

**Days 84–85: Performance Optimization**

* Topics to Learn:
  1. Caching strategies (client-side, server-side)
  2. Minimizing network requests, code splitting in React
* Assignment:
  1. Implement lazy-loading for a large React component or caching for an API route.

**Days 86–87: Final Mini Project #11 – Production-Ready**

* Project Outline:
  1. Pick one of your previous mini-projects and fully optimize it for production
  2. Add test coverage, CI/CD, environment variables, security, logging, monitoring
* Deliverable:
  1. A link to your final production-grade mini-project

**Day 88: Portfolio & Resume Review**

* Tasks:
  1. Update your portfolio site with all mini projects
  2. Polish your GitHub profile (pin top repositories, add relevant READMEs)
  3. Draft or update your resume to reflect new skills

**Weeks 12–13 (Days 89–102): Final Capstone Preparation and Kickoff**

**Day 89: Capstone Project Planning**

* Tasks:
  1. Brainstorm an idea (e-commerce platform, social media app, job board, etc.)
  2. Define user stories, features, and tech stack
  3. Create a GitHub repo and project board
* Assignment:
  1. Write a detailed specification or plan for your capstone.

**Days 90–91: Designing the Architecture**

* Tasks:
  1. Sketch database schema (MongoDB models)
  2. Outline API endpoints
  3. Plan the React front-end structure (components, pages)
* Assignment:
  1. Finalize your project scope to fit the remaining days.

**Days 92–94: Setting Up the Back-End**

* Tasks:
  1. Initialize Node/Express server with your required modules
  2. Implement basic user authentication (JWT)
  3. Establish MongoDB connection and create initial models
* Deliverable:
  1. Basic functional API with user login/signup

**Days 95–97: Front-End Scaffolding**

* Tasks:
  1. Create a React app with routes for login, signup, main dashboard
  2. Integrate authentication with your back-end
  3. Set up global state management if needed (Context/Redux)

**Days 98–100: Core Features Implementation**

* Tasks:
  1. Implement main CRUD features (e.g., posts, products, tasks)
  2. Add file uploads if needed (images for profiles, product pictures)
  3. Make sure to handle errors gracefully

**Day 101–102: Basic Testing and Refinement**

* Tasks:
  1. Write at least some unit/integration tests
  2. Conduct manual QA
  3. Prepare for final deployment

**Final 18 Days (Days 103–120): Capstone Completion & Polish**

**Days 103–105: Advanced/Optional Features**

* Possible Features:
  1. Payment integration (Stripe), or a third-party API (Google Maps, etc.)
  2. Real-time functionality with Socket.io (if relevant)
  3. Email notifications (Nodemailer)

**Days 106–110: UI/UX Polish & Performance**

* Tasks:
  1. Improve front-end design (responsive, minimal layout)
  2. Add loading states, skeletons, or spinners
  3. Optimize images and scripts

**Days 111–112: Security & Deployment**

* Tasks:
  1. Double-check JWT security, password hashing (bcrypt)
  2. Deploy your final project (front-end + back-end)
  3. Implement environment variables for production

**Days 113–115: Thorough Testing & Final Bug Fixes**

* Tasks:
  1. Beta-test your capstone with friends/peers
  2. Fix reported bugs, improve error handling
  3. Ensure performance is acceptable

**Days 116–117: Documentation & ReadMe**

* Tasks:
  1. Write a clear README with setup instructions and usage guide
  2. Document your API endpoints (Swagger or a simple markdown)

**Days 118–119: Final Presentation and Portfolio Update**

* Tasks:
  1. Record a short demo video or create a slide deck for your capstone
  2. Add the project to your portfolio website
  3. Update your LinkedIn with your new project

**Day 120: Completion & Reflection**

* Tasks:
  1. Reflect on what you’ve learned: front-end, back-end, deployment, etc.
  2. Identify areas for continued growth (DevOps, advanced testing, design patterns)
  3. Celebrate your achievement—You are now a full-stack developer!

**Additional Tips and Resources**

1. **Time Management:**
   * Aim for **1–2 hours** a day. If you have more time on weekends, you can catch up or deepen your knowledge.
2. **Project-Based Learning:**
   * Always be building. Even if you feel you “get” a concept, a small project cements your understanding.
3. **Community & Peer Learning:**
   * Consider joining a Discord group or forum for web dev learners. Teaching or discussing topics with others can accelerate your understanding.
4. **Favorite References:**
   * **MDN Web Docs** for HTML, CSS, JavaScript
   * **React Documentation** for React fundamentals and advanced concepts
   * **Node.js Docs** & **Express Docs** for back-end references
   * **MongoDB Docs** & **Mongoose** for database help
5. **Continuous Deployment & Version Control:**
   * Keep pushing every project to GitHub.
   * Use GitHub Issues and Projects to stay organized, especially for the final Capstone.
6. **Focus on Fundamentals:**
   * Make sure you deeply understand each concept—don’t just copy/paste.
   * The “why” and “how” of each technology is what will make you stand out.

**Final Note**

By following this **strict, structured 120-day roadmap**, you’ll progressively build mastery in HTML, CSS, JavaScript, React, Node.js, Express, MongoDB, and essential deployment strategies. You’ll also produce a **portfolio** of projects along the way—mini-projects to apply weekly/bi-weekly lessons, and a **Capstone Project** that ties it all together. If you remain consistent and disciplined, by **Day 120**, you will be well-prepared to take on full-stack development responsibilities and confidently showcase your skills to potential employers or clients. **Good luck and happy coding!**