100 Days – 100 Projects: Job-Ready Full-Stack Developer Challenge

A complete, project-based curriculum designed to take you from beginner to job-ready in 100 days.

PHASE 1: Web Fundamentals (Days 1-15)

Master HTML, CSS, and basic layout techniques

Day 1: Personal Portfolio Landing Page

- **Tech Stack:** HTML, CSS
- **Core Concept:** HTML semantic structure, basic CSS styling
- **Build:**
    - Create a single-page portfolio with header, about, skills, and contact sections
    - Use semantic HTML5 tags (header, nav, section, footer)
    - Apply basic CSS for typography, colors, and spacing
    - Make it clean and readable with proper hierarchy
- **Expected Outcome:** Understand HTML document structure and CSS fundamentals
- **Difficulty:** Beginner

Day 2: CSS Flexbox Photo Gallery

- **Tech Stack:** HTML, CSS (Flexbox)
- **Core Concept:** Flexbox layout system
- **Build:**
    - Create a responsive photo gallery with 12+ images
    - Use flexbox for row/column layouts
    - Implement flex-wrap, justify-content, align-items
    - Add hover effects on images
- **Expected Outcome:** Master Flexbox for one-dimensional layouts
- **Difficulty:** Beginner

Day 3: CSS Grid Dashboard Layout

- **Tech Stack:** HTML, CSS (Grid)
- **Core Concept:** CSS Grid layout system
- **Build:**
    - Design a dashboard with sidebar, header, main content, and widgets
    - Use grid-template-areas for complex layouts
    - Create responsive grid with different column counts
    - Position elements using grid lines
- **Expected Outcome:** Understand CSS Grid for two-dimensional layouts
- **Difficulty:** Beginner

Day 4: Responsive Navigation Menu

- **Tech Stack:** HTML, CSS, vanilla JavaScript
- **Core Concept:** Responsive design, mobile-first approach
- **Build:**
    - Create a navbar that works on desktop and mobile
    - Implement hamburger menu for mobile screens
    - Use media queries for breakpoints
    - Add smooth transitions for menu toggle
- **Expected Outcome:** Learn responsive design principles and basic JavaScript DOM manipulation
- **Difficulty:** Beginner

Day 5: CSS Variables Theme Switcher

- **Tech Stack:** HTML, CSS (Custom Properties), JavaScript
- **Core Concept:** CSS custom properties and dynamic styling
- **Build:**
    - Create a page with light/dark theme toggle
    - Define CSS variables for colors, fonts, spacing
    - Use JavaScript to switch between themes
    - Store user preference in localStorage
- **Expected Outcome:** Understand CSS variables and persistent data storage
- **Difficulty:** Beginner

Day 6: Form Validation Component

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Form handling and validation
- **Build:**
    - Build a registration form with email, password, confirm password fields
    - Implement real-time validation (email format, password strength)
    - Show error messages dynamically
    - Disable submit until all fields are valid
- **Expected Outcome:** Learn form validation patterns and user feedback
- **Difficulty:** Beginner

Day 7: Animated Landing Page

- **Tech Stack:** HTML, CSS (Animations)

- **Core Concept:** CSS transitions and keyframe animations
- **Build:**
    - Create a landing page with animated hero section
    - Implement fade-in, slide-in, and scale animations
    - Use @keyframes for custom animations
    - Add scroll-triggered animations
- **Expected Outcome:** Master CSS animations for better UX
- **Difficulty:** Beginner

Day 8: Pricing Table Component

- **Tech Stack:** HTML, CSS
- **Core Concept:** Component-based thinking, card layouts
- **Build:**
    - Design 3-tier pricing cards (Basic, Pro, Enterprise)
    - Highlight the recommended plan
    - Make cards responsive and stackable on mobile
    - Add hover effects and call-to-action buttons
- **Expected Outcome:** Learn to create reusable UI components
- **Difficulty:** Beginner

Day 9: FAQ Accordion

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** DOM manipulation, event handling
- **Build:**
    - Create an FAQ section with collapsible answers
    - Toggle open/close states with JavaScript
    - Animate height transitions smoothly
    - Allow only one item open at a time
- **Expected Outcome:** Understand DOM event listeners and state management basics
- **Difficulty:** Beginner

Day 10: Progress Bar Tracker

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Dynamic styling, progress tracking
- **Build:**
    - Build a multi-step form with visual progress indicator
    - Update progress bar as user completes steps
    - Validate each step before advancing
    - Add prev/next navigation
- **Expected Outcome:** Learn to track and visualize user progress
- **Difficulty:** Beginner

Day 11: Modal Dialog System

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Overlay UI patterns, focus management
- **Build:**
    - Create reusable modal component
    - Implement open/close functionality
    - Add backdrop blur and close on outside click
    - Trap focus inside modal when open
- **Expected Outcome:** Master modal patterns and accessibility basics
- **Difficulty:** Beginner

Day 12: Image Carousel/Slider

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** DOM traversal, array manipulation
- **Build:**
    - Build image slider with prev/next buttons
    - Add auto-play functionality
    - Implement dot indicators for navigation
    - Make it infinite loop
- **Expected Outcome:** Understand array indexing and timed events
- **Difficulty:** Beginner

Day 13: Countdown Timer

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Date/Time manipulation, intervals
- **Build:**
    - Create countdown to a specific date (e.g., product launch)
    - Display days, hours, minutes, seconds
    - Update every second using setInterval
    - Show "Event Started" when countdown reaches zero
- **Expected Outcome:** Learn JavaScript Date object and timers
- **Difficulty:** Beginner

Day 14: Tooltip Component

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Positioning, event handling
- **Build:**
    - Create tooltips that appear on hover
    - Position tooltips dynamically (top, bottom, left, right)
    - Handle edge cases (viewport boundaries)
    - Add smooth fade-in/out animations
- **Expected Outcome:** Master dynamic positioning and UX micro-interactions
- **Difficulty:** Beginner

Day 15: Sticky Header with Scroll Effect

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Scroll events, class toggling

- **Build:**
  - Build a header that becomes sticky on scroll
  - Change header style after scrolling past threshold
  - Add shadow and background color on scroll
  - Smooth scroll to sections on nav click
- **Expected Outcome:** Learn scroll event handling and scroll-based UI changes
- **Difficulty:** Beginner

PHASE 2: JavaScript & DOM Mastery (Days 16-30)

Deep dive into JavaScript fundamentals and DOM manipulation

Day 16: Todo List with Local Storage

- **Tech Stack:** HTML, CSS, JavaScript, localStorage
- **Core Concept:** CRUD operations, data persistence
- **Build:**
  - Add, edit, delete, and mark todos as complete
  - Store todos in localStorage
  - Filter todos (all, active, completed)
  - Clear completed todos
- **Expected Outcome:** Master localStorage and basic CRUD patterns
- **Difficulty:** Beginner

Day 17: Expense Tracker

- **Tech Stack:** HTML, CSS, JavaScript, localStorage
- **Core Concept:** Array methods, data aggregation
- **Build:**
  - Add income/expense transactions
  - Calculate and display total balance
  - List all transactions with delete option
  - Persist data in localStorage
- **Expected Outcome:** Learn array operations (map, filter, reduce) and financial calculations
- **Difficulty:** Beginner

Day 18: Quiz Application

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** State management, conditional rendering
- **Build:**
  - Display multiple-choice questions one at a time
  - Track user answers and calculate score
  - Show results screen at the end
  - Restart quiz functionality
- **Expected Outcome:** Understand application state and user flow management
- **Difficulty:** Beginner

Day 19: Weather App (API Integration)

- **Tech Stack:** HTML, CSS, JavaScript, Fetch API
- **Core Concept:** API calls, async/await, JSON
- **Build:**
  - Fetch weather data from OpenWeather API
  - Display current weather for user-entered city
  - Show temperature, humidity, wind speed, conditions
  - Handle loading states and errors
- **Expected Outcome:** Learn API integration and asynchronous JavaScript
- **Difficulty:** Intermediate

Day 20: Random Quote Generator

- **Tech Stack:** HTML, CSS, JavaScript, Fetch API
- **Core Concept:** External API consumption, randomization
- **Build:**
  - Fetch quotes from a quotes API
  - Display random quote on button click
  - Add "Tweet Quote" functionality
  - Implement loading spinner
- **Expected Outcome:** Practice API calls and social sharing features
- **Difficulty:** Beginner

Day 21: GitHub Profile Viewer

- **Tech Stack:** HTML, CSS, JavaScript, GitHub API
- **Core Concept:** REST API, dynamic data rendering
- **Build:**
  - Search for GitHub users by username
  - Display profile info (avatar, bio, repos, followers)
  - List recent repositories
  - Handle user not found scenarios
- **Expected Outcome:** Work with real-world APIs and error handling
- **Difficulty:** Intermediate

Day 22: Infinite Scroll Image Gallery

- **Tech Stack:** HTML, CSS, JavaScript, Unsplash API
- **Core Concept:** Infinite scrolling, lazy loading
- **Build:**
  - Load images from Unsplash API
  - Detect when user scrolls to bottom
  - Load more images automatically
  - Show loading indicator while fetching
- **Expected Outcome:** Implement infinite scroll pattern and pagination
- **Difficulty:** Intermediate

Day 23: Drag and Drop Task Board

- **Tech Stack:** HTML, CSS, JavaScript (Drag API)
- **Core Concept:** Drag and Drop API, event handling
- **Build:**
  - Create Kanban-style board (To Do, In Progress, Done)
  - Allow dragging tasks between columns
  - Update task status on drop
  - Persist board state in localStorage
- **Expected Outcome:** Master HTML5 Drag and Drop API
- **Difficulty:** Intermediate

Day 24: Typing Speed Test

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** String comparison, performance metrics
- **Build:**
  - Display random paragraph for typing
  - Track typing speed (WPM) and accuracy
  - Highlight incorrect characters in real-time
  - Show results with accuracy percentage
- **Expected Outcome:** Learn string manipulation and performance tracking
- **Difficulty:** Intermediate

Day 25: Color Palette Generator

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Color manipulation, clipboard API
- **Build:**
  - Generate random color palettes (5 colors)
  - Display colors with hex codes
  - Copy hex code to clipboard on click
  - Lock/unlock individual colors while generating
- **Expected Outcome:** Work with color systems and clipboard interaction
- **Difficulty:** Beginner

Day 26: Markdown Previewer

- **Tech Stack:** HTML, CSS, JavaScript, Marked.js
- **Core Concept:** Text parsing, library integration
- **Build:**
  - Create split-view editor (markdown input | preview)
  - Parse markdown to HTML in real-time
  - Support headings, lists, code blocks, links
  - Add syntax highlighting for code
- **Expected Outcome:** Integrate third-party libraries and text processing
- **Difficulty:** Intermediate

Day 27: Memory Card Game

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Game logic, array shuffling
- **Build:**
  - Create card-matching game with 8 pairs
  - Flip cards and check for matches
  - Track moves and time
  - Win condition with restart option
- **Expected Outcome:** Implement game state and logic patterns
- **Difficulty:** Intermediate

Day 28: Calculator App

- **Tech Stack:** HTML, CSS, JavaScript
- **Core Concept:** Math operations, state management
- **Build:**
  - Build functional calculator (+ - × ÷)
  - Handle decimal numbers and percentages
  - Implement clear and backspace
  - Chain multiple operations correctly
- **Expected Outcome:** Master complex state handling and edge cases
- **Difficulty:** Intermediate

Day 29: Music Player Interface

- **Tech Stack:** HTML, CSS, JavaScript, Audio API
- **Core Concept:** Web Audio API, media controls
- **Build:**
  - Play/pause audio tracks
  - Display track progress bar with seek functionality
  - Next/previous track navigation
  - Volume control slider
- **Expected Outcome:** Learn HTML5 Audio API and media handling
- **Difficulty:** Intermediate

Day 30: Recipe Finder App

- **Tech Stack:** HTML, CSS, JavaScript, Recipe API
- **Core Concept:** Search functionality, API filtering
- **Build:**
  - Search recipes by ingredients or name
  - Display recipe cards with images
  - Show detailed view on click (ingredients, instructions)
  - Save favorite recipes to localStorage
- **Expected Outcome:** Combine search, filtering, and favorites features
- **Difficulty:** Intermediate

PHASE 3: React Fundamentals (Days 31-45)

Learn React basics and component-based architecture

Day 31: React Hello World + Component Basics

- **Tech Stack:** React, Vite
- **Core Concept:** Components, JSX, props
- **Build:**
  - Set up React project with Vite
  - Create greeting component with props
  - Build reusable Card component
  - Render list of items using map()
- **Expected Outcome:** Understand React fundamentals and component composition
- **Difficulty:** Beginner

Day 32: Counter App with State

- **Tech Stack:** React, useState
- **Core Concept:** State management with hooks
- **Build:**
  - Build counter with increment/decrement buttons
  - Add reset functionality
  - Implement min/max boundaries
  - Create step size selector
- **Expected Outcome:** Master useState hook and component state
- **Difficulty:** Beginner

Day 33: Todo List in React

- **Tech Stack:** React, useState
- **Core Concept:** List rendering, event handling
- **Build:**
  - Add, delete, toggle todos
  - Filter todos by status
  - Use controlled inputs
  - Implement unique keys properly
- **Expected Outcome:** Learn React's approach to CRUD operations
- **Difficulty:** Beginner

Day 34: Form Handling in React

- **Tech Stack:** React, useState
- **Core Concept:** Controlled components, validation
- **Build:**
  - Multi-field form (name, email, password)
  - Real-time validation with error messages
  - Submit handler with console logging
  - Reset form after submission
- **Expected Outcome:** Master controlled inputs and form state
- **Difficulty:** Beginner

Day 35: useEffect - Data Fetching

- **Tech Stack:** React, useEffect, Fetch API
- **Core Concept:** Side effects, component lifecycle
- **Build:**
  - Fetch posts from JSONPlaceholder API
  - Show loading spinner during fetch
  - Display posts in cards
  - Handle errors gracefully
- **Expected Outcome:** Understand useEffect and API integration in React
- **Difficulty:** Intermediate

Day 36: Search Filter Component

- **Tech Stack:** React, useState, useEffect
- **Core Concept:** Filtering, debouncing
- **Build:**
  - Search through list of items in real-time
  - Highlight matching text
  - Show "No results" message
  - Add category filters
- **Expected Outcome:** Learn filtering patterns and search UX
- **Difficulty:** Intermediate

Day 37: Tabs Component

- **Tech Stack:** React, useState
- **Core Concept:** Conditional rendering, tab navigation
- **Build:**
  - Create tab navigation with 3+ panels
  - Switch content based on active tab
  - Style active tab indicator
  - Make component reusable with props
- **Expected Outcome:** Master conditional rendering and component reusability
- **Difficulty:** Beginner

Day 38: Modal Component in React

- **Tech Stack:** React, useState, Portal
- **Core Concept:** React Portal, prop drilling
- **Build:**
  - Create reusable modal component
  - Use React Portal for overlay
  - Close on backdrop click or Esc key
  - Pass custom content via children prop
- **Expected Outcome:** Learn React Portals and advanced component patterns
- **Difficulty:** Intermediate

Day 39: Pagination Component

- **Tech Stack:** React, useState
- **Core Concept:** Pagination logic, data slicing
- **Build:**
  - Display large dataset with pagination
  - Show 10 items per page

    - Next/Previous and page number buttons
    - Highlight current page
- **Expected Outcome:** Implement pagination from scratch
- **Difficulty:** Intermediate

Day 40: Dark Mode Toggle (Context API)

- **Tech Stack:** React, Context API, localStorage
- **Core Concept:** Global state with Context
- **Build:**
    - Create ThemeContext for dark/light mode
    - Toggle theme across entire app
    - Persist preference in localStorage
    - Apply theme to multiple components
- **Expected Outcome:** Learn Context API for global state management
- **Difficulty:** Intermediate

Day 41: Multi-Step Form

- **Tech Stack:** React, useState
- **Core Concept:** Multi-step workflows, validation
- **Build:**
    - 3-step form (personal info, address, confirmation)
    - Next/Back navigation
    - Validate each step before proceeding
    - Show progress indicator
- **Expected Outcome:** Handle complex form flows and validation
- **Difficulty:** Intermediate

Day 42: Movie Search App

- **Tech Stack:** React, useEffect, OMDB API
- **Core Concept:** API search, debouncing
- **Build:**
    - Search movies by title using OMDB API
    - Display results in grid layout
    - Show movie details on click
    - Implement search debouncing
- **Expected Outcome:** Build complete search feature with external API
- **Difficulty:** Intermediate

Day 43: Shopping Cart

- **Tech Stack:** React, useReducer/useState
- **Core Concept:** Cart logic, state updates
- **Build:**
    - Add/remove items to cart
    - Update quantities
    - Calculate total price
    - Display cart badge with item count
- **Expected Outcome:** Manage complex state with cart operations
- **Difficulty:** Intermediate

Day 44: Custom Hooks - useLocalStorage

- **Tech Stack:** React, Custom Hooks
- **Core Concept:** Custom hook creation, logic reuse
- **Build:**
    - Create useLocalStorage hook
    - Sync state with localStorage automatically
    - Use hook in 2-3 different components
    - Handle JSON serialization
- **Expected Outcome:** Learn to extract and reuse logic with custom hooks
- **Difficulty:** Intermediate

Day 45: Notes App with Categories

- **Tech Stack:** React, useState, localStorage
- **Core Concept:** CRUD + categorization
- **Build:**
    - Create, edit, delete notes
    - Assign categories to notes
    - Filter notes by category
    - Search notes by content
    - Persist everything in localStorage
- **Expected Outcome:** Combine multiple features into cohesive app
- **Difficulty:** Intermediate

PHASE 4: Advanced React + Styling (Days 46-60)

Master advanced React patterns and modern styling

Day 46: Tailwind CSS Setup + Responsive Dashboard

- **Tech Stack:** React, Tailwind CSS
- **Core Concept:** Utility-first CSS, responsive design
- **Build:**
    - Set up Tailwind in React project
    - Build dashboard with sidebar and cards
    - Make fully responsive with Tailwind breakpoints
    - Use Tailwind's color system and spacing scale
- **Expected Outcome:** Master Tailwind CSS fundamentals
- **Difficulty:** Beginner

Day 47: Component Library with Tailwind

- **Tech Stack:** React, Tailwind CSS
- **Core Concept:** Reusable components, design system
- **Build:**
    - Create Button, Card, Badge, Alert components
    - Support variants (primary, secondary, danger)

    - Make components fully configurable via props
    - Document usage of each component
- **Expected Outcome:** Build reusable component library
- **Difficulty:** Intermediate

Day 48: React Router - Multi-Page App

- **Tech Stack:** React, React Router
- **Core Concept:** Client-side routing, navigation
- **Build:**
    - Set up routes for Home, About, Contact, Blog
    - Create navigation bar with active link styling
    - Implement nested routes for blog posts
    - Add 404 Not Found page
- **Expected Outcome:** Master client-side routing in React
- **Difficulty:** Intermediate

Day 49: Protected Routes + Auth Simulation

- **Tech Stack:** React, React Router, Context API
- **Core Concept:** Route protection, conditional rendering
- **Build:**
    - Create login/logout flow (simulated)
    - Protect dashboard route (requires "login")
    - Redirect to login if not authenticated
    - Store auth state in Context
- **Expected Outcome:** Implement authentication flow and route guards
- **Difficulty:** Intermediate

Day 50: Infinite Scroll with React

- **Tech Stack:** React, useEffect, Intersection Observer
- **Core Concept:** Performance optimization, lazy loading
- **Build:**
    - Load paginated data from API
    - Detect scroll to bottom with Intersection Observer
    - Load more items automatically
    - Show loading state
- **Expected Outcome:** Implement performant infinite scroll
- **Difficulty:** Intermediate

Day 51: Virtualized List (React Window)

- **Tech Stack:** React, react-window
- **Core Concept:** Virtual scrolling, performance
- **Build:**
    - Render large list (1000+ items) efficiently
    - Use react-window for virtualization
    - Compare performance with/without virtualization
    - Implement dynamic item heights
- **Expected Outcome:** Optimize rendering for large datasets
- **Difficulty:** Advanced

Day 52: Form with React Hook Form

- **Tech Stack:** React, React Hook Form, Zod
- **Core Concept:** Form libraries, schema validation
- **Build:**
    - Build complex form with React Hook Form
    - Add validation with Zod schema
    - Handle nested fields and arrays
    - Display validation errors properly
- **Expected Outcome:** Use industry-standard form library
- **Difficulty:** Intermediate

Day 53: Data Table with Sorting & Filtering

- **Tech Stack:** React, Tailwind
- **Core Concept:** Table logic, data manipulation
- **Build:**
    - Display data in sortable table
    - Sort by column (ascending/descending)
    - Filter rows by search query
    - Highlight selected rows
- **Expected Outcome:** Build interactive data tables
- **Difficulty:** Intermediate

Day 54: Charts Dashboard

- **Tech Stack:** React, Recharts/Chart.js
- **Core Concept:** Data visualization
- **Build:**
    - Create dashboard with 3+ chart types
    - Line chart, bar chart, pie chart
    - Fetch data from API or use mock data
    - Make charts responsive
- **Expected Outcome:** Integrate charting libraries and visualize data
- **Difficulty:** Intermediate

Day 55: Real-Time Search with Debouncing

- **Tech Stack:** React, Custom Hook, API
- **Core Concept:** Performance optimization, debouncing
- **Build:**
    - Search API as user types
    - Implement debounce hook (500ms delay)
    - Show search suggestions dropdown
    - Cancel previous requests
- **Expected Outcome:** Optimize API calls with debouncing
- **Difficulty:** Intermediate

Day 56: Image Upload with Preview

- **Tech Stack:** React, FileReader API
- **Core Concept:** File handling, image preview
- **Build:**
    - Upload image with drag-and-drop or file input

    - Show image preview before upload
    - Validate file type and size
    - Display multiple images
- **Expected Outcome:** Handle file uploads in React
- **Difficulty:** Intermediate

Day 57: Notification/Toast System

- **Tech Stack:** React, Context API, Tailwind
- **Core Concept:** Global notifications, animations
- **Build:**
    - Create toast notification system
    - Support success, error, warning, info types
    - Auto-dismiss after timeout
    - Stack multiple notifications
- **Expected Outcome:** Build reusable notification system
- **Difficulty:** Intermediate

Day 58: Optimistic UI Updates

- **Tech Stack:** React, useState, API
- **Core Concept:** UX optimization, rollback on error
- **Build:**
    - Like button with optimistic update
    - Update UI immediately, then call API
    - Rollback if API fails
    - Show error message on failure
- **Expected Outcome:** Implement optimistic UI pattern
- **Difficulty:** Advanced

Day 59: useReducer - Complex State Management

- **Tech Stack:** React, useReducer
- **Core Concept:** Reducer pattern, action dispatching
- **Build:**
    - Refactor shopping cart to use useReducer
    - Define actions (ADD, REMOVE, UPDATE_QUANTITY)
    - Handle complex state transitions
    - Compare with useState approach
- **Expected Outcome:** Master useReducer for complex state
- **Difficulty:** Intermediate

Day 60: Error Boundary Component

- **Tech Stack:** React, Error Boundaries
- **Core Concept:** Error handling, graceful degradation
- **Build:**
    - Create Error Boundary component
    - Catch errors in child components
    - Display fallback UI on error
    - Log errors to console
- **Expected Outcome:** Implement error handling best practices
- **Difficulty:** Intermediate

PHASE 5: Backend Fundamentals (Days 61-75)

Learn Node.js, Express, and API development

Day 61: Node.js Basics - File System Operations

- **Tech Stack:** Node.js, fs module
- **Core Concept:** File I/O, async operations
- **Build:**
    - Read and write text files
    - Create, rename, delete files
    - Read directory contents
    - Use both sync and async methods
- **Expected Outcome:** Understand Node.js file system operations
- **Difficulty:** Beginner

Day 62: Express Server Setup

- **Tech Stack:** Node.js, Express
- **Core Concept:** HTTP servers, routing basics
- **Build:**
    - Set up basic Express server
    - Create 3-4 routes (GET, POST)
    - Serve static HTML files
    - Add basic logging middleware
- **Expected Outcome:** Learn Express fundamentals and middleware
- **Difficulty:** Beginner

Day 63: REST API - CRUD with In-Memory Storage

- **Tech Stack:** Node.js, Express
- **Core Concept:** RESTful design, HTTP methods
- **Build:**
    - Create REST API for tasks (array in memory)
    - GET /tasks, POST /tasks, PUT /tasks/:id, DELETE /tasks/:id
    - Use proper status codes (200, 201, 404)
    - Test with Postman/Thunder Client
- **Expected Outcome:** Build RESTful APIs with proper conventions
- **Difficulty:** Intermediate

Day 64: Middleware Deep Dive

- **Tech Stack:** Node.js, Express
- **Core Concept:** Middleware chain, request processing
- **Build:**
    - Create custom logging middleware
    - Add error handling middleware
    - Implement request validation middleware

    - Use built-in middleware (express.json)
- **Expected Outcome:** Master Express middleware pattern
- **Difficulty:** Intermediate

Day 65: Environment Variables & Config

- **Tech Stack:** Node.js, Express, dotenv
- **Core Concept:** Configuration management, security
- **Build:**
    - Use dotenv for environment variables
    - Store API keys and ports in .env
    - Create different configs for dev/prod
    - Never commit .env file
- **Expected Outcome:** Manage configurations securely
- **Difficulty:** Beginner

Day 66: File Upload API

- **Tech Stack:** Node.js, Express, Multer
- **Core Concept:** Multipart form data, file handling
- **Build:**
    - Create endpoint for file uploads
    - Use Multer middleware
    - Validate file types and sizes
    - Save files to uploads folder
- **Expected Outcome:** Handle file uploads in backend
- **Difficulty:** Intermediate

Day 67: Input Validation with Joi

- **Tech Stack:** Node.js, Express, Joi
- **Core Concept:** Request validation, data integrity
- **Build:**
    - Add Joi validation to API endpoints
    - Validate request body, params, query
    - Return detailed validation errors
    - Create reusable validation middleware
- **Expected Outcome:** Implement robust input validation
- **Difficulty:** Intermediate

Day 68: Error Handling Best Practices

- **Tech Stack:** Node.js, Express
- **Core Concept:** Error handling patterns
- **Build:**
    - Create custom error classes
    - Implement global error handler
    - Handle async errors with try-catch
    - Return consistent error responses
- **Expected Outcome:** Master error handling in Express
- **Difficulty:** Intermediate

Day 69: API Rate Limiting

- **Tech Stack:** Node.js, Express, express-rate-limit
- **Core Concept:** Rate limiting, DDoS protection
- **Build:**
    - Add rate limiting to API endpoints
    - Limit requests per IP (100 req/15min)
    - Return 429 status when exceeded
    - Configure different limits per endpoint
- **Expected Outcome:** Implement API security with rate limiting
- **Difficulty:** Intermediate

Day 70: CORS Configuration

- **Tech Stack:** Node.js, Express, cors
- **Core Concept:** Cross-Origin Resource Sharing
- **Build:**
    - Enable CORS for API
    - Allow specific origins only
    - Configure allowed methods and headers
    - Handle preflight requests
- **Expected Outcome:** Understand and configure CORS properly
- **Difficulty:** Intermediate

Day 71: MongoDB Setup & Connection

- **Tech Stack:** Node.js, MongoDB, Mongoose
- **Core Concept:** NoSQL databases, ODM
- **Build:**
    - Set up MongoDB Atlas account
    - Connect to MongoDB with Mongoose
    - Create first schema and model
    - Test connection and basic queries
- **Expected Outcome:** Connect Node.js to MongoDB
- **Difficulty:** Beginner

Day 72: MongoDB CRUD Operations

- **Tech Stack:** Node.js, Express, MongoDB, Mongoose
- **Core Concept:** Database operations
- **Build:**
    - Replace in-memory storage with MongoDB
    - Implement full CRUD with Mongoose
    - Add data validation in schema
    - Handle database errors
- **Expected Outcome:** Perform database operations with Mongoose
- **Difficulty:** Intermediate

Day 73: Data Relationships in MongoDB

- **Tech Stack:** Node.js, MongoDB, Mongoose
- **Core Concept:** Document relationships, references
- **Build:**
    - Create User and Post schemas
    - Implement one-to-many relationship
    - Use populate() for referenced data
    - Query related documents

- **Expected Outcome:** Model and query related data
- **Difficulty:** Intermediate

Day 74: Pagination & Filtering API

- **Tech Stack:** Node.js, Express, MongoDB
- **Core Concept:** Query optimization, pagination
- **Build:**
  - Add pagination to GET endpoints
  - Support ?page=2&limit=10 queries
  - Implement filtering by fields
  - Return total count and pages
- **Expected Outcome:** Build scalable query endpoints
- **Difficulty:** Intermediate

Day 75: Search Functionality

- **Tech Stack:** Node.js, Express, MongoDB
- **Core Concept:** Text search, regex queries
- **Build:**
  - Add search endpoint with text queries
  - Use MongoDB text indexes
  - Implement regex-based search
  - Sort results by relevance
- **Expected Outcome:** Implement full-text search
- **Difficulty:** Intermediate

PHASE 6: Full-Stack Integration (Days 76-85)

Connect frontend and backend into complete applications

Day 76: Full-Stack Todo App

- **Tech Stack:** React, Node.js, Express, MongoDB
- **Core Concept:** Frontend-backend integration
- **Build:**
  - Build REST API for todos
  - Create React frontend consuming API
  - Connect with fetch/axios
  - Handle loading and error states
- **Expected Outcome:** Build your first full-stack CRUD app
- **Difficulty:** Intermediate

Day 77: Blog Platform (Posts & Comments)

- **Tech Stack:** React, Node.js, Express, MongoDB
- **Core Concept:** Nested resources, relationships
- **Build:**
  - Create posts and comments APIs
  - Build blog frontend with post list/detail
  - Add comment section to posts
  - Implement create/delete for both
- **Expected Outcome:** Handle nested API resources
- **Difficulty:** Intermediate

Day 78: User Registration & Login

- **Tech Stack:** React, Node.js, Express, MongoDB, bcrypt
- **Core Concept:** Authentication basics, password hashing
- **Build:**
  - Create user registration endpoint
  - Hash passwords with bcrypt
  - Implement login with password verification
  - Build login/register forms in React
- **Expected Outcome:** Implement basic authentication
- **Difficulty:** Intermediate

Day 79: JWT Authentication

- **Tech Stack:** Node.js, Express, JWT, React
- **Core Concept:** Token-based authentication
- **Build:**
  - Generate JWT on successful login
  - Verify JWT in protected routes
  - Store token in React (localStorage/state)
  - Send token in Authorization header
- **Expected Outcome:** Implement JWT authentication flow
- **Difficulty:** Advanced

Day 80: Protected Routes & Authorization

- **Tech Stack:** React, Node.js, Express, JWT
- **Core Concept:** Authorization, role-based access
- **Build:**
  - Add user roles (user, admin)
  - Protect endpoints based on roles
  - Implement frontend route guards
  - Redirect unauthenticated users
- **Expected Outcome:** Master authorization patterns
- **Difficulty:** Advanced

Day 81: Image Upload Full-Stack

- **Tech Stack:** React, Node.js, Express, Multer, MongoDB
- **Core Concept:** File uploads end-to-end
- **Build:**
  - Upload images from React
  - Store files on server with Multer
  - Save file paths in MongoDB
  - Display uploaded images
- **Expected Outcome:** Handle file uploads across stack
- **Difficulty:** Intermediate

Day 82: Real-Time Chat with Socket.io

- **Tech Stack:** React, Node.js, Socket.io
- **Core Concept:** WebSockets, real-time communication
- **Build:**
    - Set up Socket.io server
    - Connect React client to socket
    - Send/receive messages in real-time
    - Display online users
- **Expected Outcome:** Build real-time features with WebSockets
- **Difficulty:** Advanced

Day 83: E-commerce Product Catalog

- **Tech Stack:** React, Node.js, Express, MongoDB
- **Core Concept:** Complex data models, filtering
- **Build:**
    - Create products API with categories
    - Build product listing page
    - Implement filters (category, price range)
    - Add product detail page
- **Expected Outcome:** Build product catalog with filtering
- **Difficulty:** Intermediate

Day 84: Shopping Cart Full-Stack

- **Tech Stack:** React, Node.js, Express, MongoDB
- **Core Concept:** Session management, cart logic
- **Build:**
    - Store cart in database per user
    - Add/remove/update cart items via API
    - Sync React cart state with backend
    - Calculate totals on server
- **Expected Outcome:** Implement persistent shopping cart
- **Difficulty:** Advanced

Day 85: Payment Integration Simulation

- **Tech Stack:** React, Node.js, Express
- **Core Concept:** Payment flow, webhooks (simulated)
- **Build:**
    - Create checkout endpoint
    - Simulate payment processing
    - Send order confirmation
    - Update order status
    - (Use test mode, no real payments)
- **Expected Outcome:** Understand payment flow architecture
- **Difficulty:** Advanced

PHASE 7: Performance & Production (Days 86-92)

Optimize, secure, and prepare apps for production

Day 86: API Response Caching

- **Tech Stack:** Node.js, Express, node-cache/Redis
- **Core Concept:** Caching strategies, performance
- **Build:**
    - Add caching middleware to API
    - Cache expensive database queries
    - Set cache expiration times
    - Invalidate cache on data updates
- **Expected Outcome:** Improve API performance with caching
- **Difficulty:** Advanced

Day 87: Database Indexing & Query Optimization

- **Tech Stack:** MongoDB, Mongoose
- **Core Concept:** Performance optimization, indexes
- **Build:**
    - Analyze slow queries with explain()
    - Add indexes to frequently queried fields
    - Optimize aggregation pipelines
    - Measure performance improvements
- **Expected Outcome:** Optimize database performance
- **Difficulty:** Advanced

Day 88: React Performance Optimization

- **Tech Stack:** React, React.memo, useMemo, useCallback
- **Core Concept:** Memoization, re-render optimization
- **Build:**
    - Identify unnecessary re-renders with DevTools
    - Use React.memo for expensive components
    - Apply useMemo and useCallback
    - Measure performance gains
- **Expected Outcome:** Optimize React app performance
- **Difficulty:** Advanced

Day 89: Code Splitting & Lazy Loading

- **Tech Stack:** React, React Router, dynamic imports
- **Core Concept:** Bundle optimization, lazy loading
- **Build:**
    - Implement route-based code splitting
    - Lazy load components with React.lazy
    - Add loading fallbacks
    - Analyze bundle sizes
- **Expected Outcome:** Reduce initial bundle size
- **Difficulty:** Advanced

Day 90: Security Headers & HTTPS

- **Tech Stack:** Node.js, Express, Helmet
- **Core Concept:** Web security, headers

- **Build:**
  - Add Helmet middleware for security headers
  - Configure CSP (Content Security Policy)
  - Set secure cookies (httpOnly, secure)
  - Prepare for HTTPS deployment
- **Expected Outcome:** Implement security best practices
- **Difficulty:** Intermediate

Day 91: Input Sanitization & XSS Prevention

- **Tech Stack:** Node.js, Express, validator, DOMPurify
- **Core Concept:** Security, injection prevention
- **Build:**
  - Sanitize user inputs on backend
  - Prevent XSS attacks in React
  - Validate and escape HTML content
  - Test with malicious inputs
- **Expected Outcome:** Secure app against common attacks
- **Difficulty:** Advanced

Day 92: Logging & Monitoring Setup

- **Tech Stack:** Node.js, Winston/Morgan, PM2
- **Core Concept:** Application monitoring, debugging
- **Build:**
  - Set up structured logging with Winston
  - Log errors, warnings, and info
  - Rotate log files
  - Monitor app with PM2
- **Expected Outcome:** Implement production logging
- **Difficulty:** Intermediate

PHASE 8: Deployment & DevOps (Days 93-96)

Deploy applications to production

Day 93: Deploy Backend to Render/Railway

- **Tech Stack:** Node.js, Express, MongoDB, Render
- **Core Concept:** Backend deployment, environment config
- **Build:**
  - Deploy Express API to Render/Railway
  - Connect to MongoDB Atlas
  - Configure environment variables
  - Test deployed endpoints
- **Expected Outcome:** Deploy backend to production
- **Difficulty:** Intermediate

Day 94: Deploy Frontend to Vercel/Netlify

- **Tech Stack:** React, Vercel/Netlify
- **Core Concept:** Frontend deployment, build optimization
- **Build:**
  - Build production React app
  - Deploy to Vercel or Netlify
  - Configure environment variables
  - Set up custom domain (optional)
- **Expected Outcome:** Deploy frontend to production
- **Difficulty:** Beginner

Day 95: CI/CD Pipeline with GitHub Actions

- **Tech Stack:** GitHub Actions, Node.js
- **Core Concept:** Continuous integration/deployment
- **Build:**
  - Create GitHub Actions workflow
  - Run tests on every push
  - Auto-deploy on main branch merge
  - Add build status badge
- **Expected Outcome:** Automate testing and deployment
- **Difficulty:** Advanced

Day 96: Docker Containerization

- **Tech Stack:** Docker, Node.js, MongoDB
- **Core Concept:** Containerization, portability
- **Build:**
  - Create Dockerfile for backend
  - Set up docker-compose with MongoDB
  - Run app in containers
  - Push image to Docker Hub
- **Expected Outcome:** Containerize applications with Docker
- **Difficulty:** Advanced

PHASE 9: Advanced Features (Days 97-100)

Modern features and final portfolio projects

Day 97: AI Integration - OpenAI API

- **Tech Stack:** Node.js, Express, OpenAI API, React
- **Core Concept:** AI/ML integration, API orchestration
- **Build:**
  - Create AI-powered feature (text generation/chat)
  - Call OpenAI API from backend
  - Build React UI for AI interaction
  - Handle streaming responses
- **Expected Outcome:** Integrate AI capabilities into apps
- **Difficulty:** Advanced

Day 98: Email Notifications with Nodemailer

- **Tech Stack:** Node.js, Express, Nodemailer
- **Core Concept:** Email automation, transactional emails
- **Build:**
  - Send welcome email on registration
  - Send password reset emails
  - Use email templates
  - Configure SMTP settings
- **Expected Outcome:** Implement email functionality
- **Difficulty:** Intermediate

Day 99: Full-Stack Social Media Dashboard

- **Tech Stack:** React, Node.js, Express, MongoDB, JWT, Socket.io
- **Core Concept:** Complex full-stack application
- **Build:**
  - User profiles with avatar upload
  - Create, like, comment on posts
  - Follow/unfollow users
  - Real-time notifications
  - Feed with pagination
- **Expected Outcome:** Build complex production-ready app
- **Difficulty:** Advanced

Day 100: Portfolio Website Showcase

- **Tech Stack:** React, Tailwind, React Router
- **Core Concept:** Professional presentation, SEO
- **Build:**
  - Build professional portfolio showcasing all 100 projects
  - Add project cards with live demos and code links
  - Implement contact form
  - Add resume/CV section
  - Optimize for SEO and performance
  - Deploy to custom domain
- **Expected Outcome:** Create job-application-ready portfolio
- **Difficulty:** Intermediate

WHAT YOU'VE ACHIEVED

After completing these 100 projects, you will have:

✅ **Built 100+ real projects** hosted on GitHub
✅ **Mastered** HTML, CSS, JavaScript, React, Node.js, Express, MongoDB
✅ **Learned** authentication, authorization, API design, database modeling
✅ **Implemented** security best practices, performance optimization
✅ **Deployed** full-stack applications to production
✅ **Integrated** third-party APIs, AI services, real-time features
✅ **Created** a portfolio that proves you can build production-ready applications

You are now ready to apply for internships and junior developer positions.

NEXT STEPS

1. **Polish your GitHub** - Ensure each project has a good README
2. **Update your resume** - Highlight key projects with tech stacks
3. **Practice interviews** - Review concepts you learned
4. **Start applying** - You have the skills companies are looking for
5. **Keep building** - Continue learning by building more complex projects

Good luck! 🚀