

# 20-Week Full-Stack Developer Curriculum for Canadian Job Market

## Structured Plan for Career Changers with Depression Support

---

### Program philosophy and structure

This curriculum eliminates decision paralysis through pre-planned daily structures while maintaining flexibility for depression management. [Let's learn about +3 ↗](#) **You are a developer from day one**—not "aspiring" or "learning." This identity shift is critical for motivation maintenance over 20 weeks.

### Core principles

- Structure over motivation:** Your schedule operates independently of how you feel. On low-energy days, you follow a reduced template (provided below), not ad-hoc decisions.
  - 50-60% understanding threshold:** You'll start building projects when you understand concepts at 50-60%, not 100%. This prevents perfectionism and tutorial hell while accelerating real learning through problem-solving. [DEV Community ↗](#)
  - Interleaving for retention:** Daily schedules alternate between frontend, backend, and algorithms in 50-90 minute blocks with mandatory breaks. Research shows this improves retention by 76% after one month compared to blocked learning. [Effectiviology ↗](#) [Scientific American ↗](#)
  - Progress over perfection:** You'll track completion, not mastery. Completionism builds momentum; perfectionism creates paralysis.
- 

### Daily schedule templates

#### Standard day template (Monday-Thursday, Weeks 1-16)

**Total: 10-12 hours study, 2-3 hours breaks**



6:00-7:30 AM Morning routine (exercise 20 min, breakfast, preparation)

8:00-9:50 AM BLOCK 1: Topic A focused study (50 min work, 10 min break, 50 min work)

9:50-10:10 AM Break (walk, stretch, hydrate - no screens)

10:10-12:00 PM BLOCK 2: Topic B focused study (50 min, 10 min break, 50 min)

12:00-12:30 PM Lunch + extended break

12:30-2:20 PM BLOCK 3: Topic C focused study (coding practice or project)

2:20-2:40 PM Break (exercise recommended - optimal for memory consolidation)

2:40-4:30 PM BLOCK 4: Project work (apply morning concepts)

4:30-5:00 PM Extended break + snack

5:00-6:50 PM BLOCK 5: LeetCode / Algorithm practice

6:50-7:10 PM Break

7:10-8:30 PM BLOCK 6: Review, documentation, planning tomorrow

8:30-10:30 PM Wind down, dinner, evening routine, sleep prep

10:30 PM Sleep (target 7-8 hours)

Reduced intensity day (Friday or low-energy days)

Total: 6-8 hours study



8:00-12:00 PM Morning blocks (concepts and light practice)

12:00-1:00 PM Extended lunch

1:00-4:00 PM Project work or review (gentler cognitive load)

4:00 PM+ Free time (social, hobbies, rest)

Depression management day (use when needed)

Total: 3-4 hours minimum



Use 15-minute Pomodoros instead of 50-minute blocks

Focus: One small task = victory

- Open VS Code and write 5 lines = WIN

- Watch one tutorial = WIN

- Solve one Easy LeetCode = WIN

- Review yesterday's notes = WIN

Non-negotiable: Take breaks, eat meals, get outside once

## Weekend structure

**Saturday:** 4-6 hours light study (personal project exploration, optional) OR complete rest

**Sunday:** 2-4 hours max (weekly review, meal prep, planning), rest and recovery

**Every 3-4 weeks:** Take 2-3 consecutive days completely off

---

## 20-Week curriculum breakdown

### Phase 1: Foundations (Weeks 1-6)

**Learning objectives:** Build fundamental skills in HTML/CSS/JavaScript, Git, basic command line, and programming logic. Establish sustainable study habits and routines.

#### Week 1: Environment setup and JavaScript fundamentals

**Primary objective:** Establish development environment and core JavaScript concepts

**Daily topic rotation:**

- Topic A: JavaScript basics (variables, types, operators)
- Topic B: HTML/CSS fundamentals
- Topic C: Git basics and command line






**Daily schedule:**

- 8:00-10:00: JavaScript ES6+ syntax (let/const, template literals, destructuring)
- 10:10-12:00: HTML5 semantic structure + CSS basics
- 12:30-2:20: Git setup, basic commands (init, add, commit, push)
- 2:40-4:30: Practice project: Build simple HTML pages with styling
- 5:00-6:50: LeetCode Easy problems (arrays, strings) - solve 1-2 problems
- 7:10-8:30: Documentation of learnings, setup portfolio repository

**Resources:**

- JavaScript: Eloquent JavaScript (Chapters 1-4) or JavaScript.info
- HTML/CSS: MDN Web Docs HTML/CSS guides
- Git: Git official tutorial + MIT Missing Semester (Version Control)
- LeetCode: Start with Two Sum, Contains Duplicate

**Milestones:**

-  VS Code setup with extensions (ESLint, Prettier, GitLens)
-  GitHub account created, first repository pushed
-  Can explain let vs const, template literals, arrow functions
-  Built 3 simple HTML/CSS pages
-  Solved 5 LeetCode Easy problems

**Move-on criteria:** Can write basic JavaScript functions, create HTML structure, commit to Git without looking up commands

**AI tool usage this week:**

- Perplexity: Research concepts when documentation is unclear
  - ChatGPT: Get alternative explanations for confusing concepts
  - **NO AI for:** Writing code solutions or debugging. Struggle builds fundamentals.
- 

**Week 2: JavaScript intermediate and DOM manipulation**

**Primary objective:** Master JavaScript functions, loops, DOM manipulation

**Daily topic rotation:**

- Topic A: Functions, scope, closures
- Topic B: Arrays, objects, iteration methods
- Topic C: DOM manipulation and events





**Daily schedule:**

- 8:00-10:00: Functions (regular, arrow, callbacks, closures)
- 10:10-12:00: Array methods (map, filter, reduce, forEach)
- 12:30-2:20: DOM selection, manipulation, event listeners
- 2:40-4:30: Practice project: Interactive TODO list (vanilla JS)
- 5:00-6:50: LeetCode Easy (two pointers, hash maps) - 2 problems
- 7:10-8:30: Code review of own work, planning

**Resources:**

- JavaScript.info (Functions, Objects chapters)
- MDN: DOM manipulation guide
- LeetCode: Valid Palindrome, Two Sum II

**Milestones:**

-  TODO list with add, delete, mark complete functionality
-  Understands callbacks and closures conceptually
-  Can manipulate DOM without jQuery
-  Solved 10 cumulative LeetCode Easy problems

**Move-on criteria:** Can build interactive webpage with event handlers without tutorial

---

**Week 3: Asynchronous JavaScript and APIs**

**Primary objective:** Understand promises, async/await, fetch API

**Daily topic rotation:**

- Topic A: Asynchronous JS (callbacks, promises, async/await)
- Topic B: Fetch API and working with JSON
- Topic C: Error handling and debugging





**Daily schedule:**

- 8:00-10:00: Promises fundamentals and chaining
- 10:10-12:00: Async/await syntax and error handling
- 12:30-2:20: Fetch API, working with REST APIs
- 2:40-4:30: Project: Weather app using OpenWeather API
- 5:00-6:50: LeetCode Easy/beginning Medium (sliding window) - 2 problems
- 7:10-8:30: Debugging practice, code review

**Resources:**

- JavaScript.info (Async chapter)
- MDN: Using Fetch
- Free APIs: OpenWeather, JSONPlaceholder for practice

**Milestones:**

-  Weather app fetches and displays live data
-  Understands promise chains vs async/await
-  Can handle errors in async code
-  15+ LeetCode Easy problems solved

**Move-on criteria:** Can consume API and handle responses without looking up syntax every time

---

**Week 4: React fundamentals**

**Primary objective:** Understand components, JSX, props, basic hooks

**Daily topic rotation:**

- Topic A: React components and JSX
- Topic B: Props and state (useState)
- Topic C: useEffect and component lifecycle





**Daily schedule:**

- 8:00-10:00: Create React App setup, components, JSX syntax
- 10:10-12:00: Props passing, component composition
- 12:30-2:20: useState hook, state management basics
- 2:40-4:30: Project: Convert TODO list to React
- 5:00-6:50: LeetCode Easy/Medium mix (stacks, queues) - 2 problems
- 7:10-8:30: Review React docs, planning component architecture

**Resources:**

- Official React docs (new react.dev, excellent for learning)
- Jonas Schmedtmann React course (Udemy, first sections)
- Full Stack Open Part 1

**Milestones:**

-  React TODO list with components, state, props
-  Understands component reusability
-  Can use useState confidently
-  20+ LeetCode problems solved (15 Easy, 5 Medium attempted)

**Move-on criteria:** Can create React app with multiple components sharing state

---

**Week 5-6: React intermediate and first portfolio project**

**Primary objective:** Complete Pathfinding Visualizer (Portfolio Project #1)

**Week 5 daily rotation:**

- Topic A: React hooks (useEffect, useCallback, useMemo)
- Topic B: Algorithm visualization concepts
- Topic C: Canvas API or grid rendering

**Week 6: Full project implementation**

**Project: Pathfinding Visualizer**

- Visualize Dijkstra's, A\*, BFS, DFS algorithms
- Interactive grid where users add walls
- Animate the algorithm's exploration






**Daily schedule (adjusted for project focus):**

- 8:00-10:00: Algorithm study (implement algorithms in JavaScript)
- 10:10-12:00: React component architecture for project
- 12:30-4:30: Project implementation (longer blocks for flow state)
- 5:00-6:50: LeetCode Medium (graphs, BFS/DFS) - 1-2 problems
- 7:10-8:30: Documentation, README writing

**Resources:**

- Clement Samuel's Pathfinding Visualizer tutorial (concept only, build yourself)
- Algorithm visualizer examples for inspiration
- React Hook documentation

**Milestones:**

-  Functional pathfinding visualizer deployed
-  At least 2 algorithms working (Dijkstra + BFS minimum)
-  Professional README with challenges overcome
-  Deployed to Netlify or Vercel
-  30+ LeetCode problems total (20 Easy, 10 Medium)

**Move-on criteria:** Project is **complete and deployed**—not perfect. You can demo it in 5 minutes.

**AI tool usage Weeks 1-6:**

- Perplexity: Research algorithms, React patterns
- ChatGPT: Explain concepts when stuck, ask for alternative approaches

- **NO AI for:** Writing project code or LeetCode solutions. Implementation must be yours.
- Use AI as tutor: "Explain why useEffect runs twice in dev mode" not "Write my useEffect"

## CHECKPOINT: End of Week 6

- Can build React applications independently
- Solved 30+ LeetCode problems (success rate: 80%+ Easy, 40%+ Medium)
- 1 portfolio project deployed
- Comfortable with Git workflow

### Red flags:

- Can't build React app without tutorial → Rebuild TODO list from scratch
  - Less than 25 LeetCode solved → Dedicate Week 7 to algorithms before continuing
  - Project incomplete → Simplify scope, finish in Week 7
- 

## Phase 2: Backend and databases (Weeks 7-12)

**Learning objectives:** Master Node.js, Express, PostgreSQL, REST API design, authentication, and complete second portfolio project.

### Week 7: Node.js and Express fundamentals

**Primary objective:** Understand server-side JavaScript, Express setup, routing

#### Daily topic rotation:

- Topic A: Node.js fundamentals (modules, npm, async patterns)
- Topic B: Express server setup and routing
- Topic C: Middleware and error handling

#### Daily schedule:

- 8:00-10:00: Node.js modules, require vs import, package.json
- 10:10-12:00: Express basics (routes, app structure, middleware)
- 12:30-2:20: Creating REST API endpoints (GET, POST, PUT, DELETE)
- 2:40-4:30: Build simple Express server with multiple routes
- 5:00-6:50: LeetCode Medium (trees, binary search) - 1-2 problems
- 7:10-8:30: Review Express patterns, API design principles

#### Resources:

- Official Node.js documentation
- Express documentation
- Auth0 "Node.js and TypeScript Tutorial: Build a CRUD API"
- LogRocket "Express with TypeScript" (2025 setup)

#### Milestones:

- ☒ Express server running with 5+ endpoints
- ☒ Understands middleware pattern
- ☒ Can handle errors properly
- ☒ 35+ LeetCode problems (including binary trees)

**Move-on criteria:** Can create basic REST API from scratch

---

## Week 8: PostgreSQL and database fundamentals

**Primary objective:** Learn SQL, database design, connect Node.js to PostgreSQL

**Daily topic rotation:**

- Topic A: SQL basics (SELECT, INSERT, UPDATE, DELETE)
- Topic B: Database design (tables, relationships, normalization)
- Topic C: Node.js + PostgreSQL integration (node-postgres)





**Daily schedule:**

- 8:00-10:00: PostgreSQL installation, SQL basics
- 10:10-12:00: Database schema design, foreign keys, joins
- 12:30-2:20: Connecting Express to PostgreSQL
- 2:40-4:30: Build API with database persistence
- 5:00-6:50: LeetCode Medium (hash tables, heaps) - 1-2 problems
- 7:10-8:30: Schema design practice, documenting database

**Resources:**

- PostgreSQLTutorial.com (comprehensive free resource) [Neon](#) ↗
- Official PostgreSQL documentation
- node-postgres library documentation

**Milestones:**

-  Created database with 3+ related tables
-  Express API connects to PostgreSQL
-  Can write JOIN queries
-  40+ LeetCode problems

**Move-on criteria:** Can design schema and query database for basic applications

---

## Week 9: TypeScript introduction

**Primary objective:** Add TypeScript to existing skills (frontend and backend)

**Daily topic rotation:**

- Topic A: TypeScript basics (types, interfaces, generics)
- Topic B: TypeScript with React
- Topic C: TypeScript with Node.js/Express

**Daily schedule:**





- 8:00-10:00: TypeScript fundamentals (types, interfaces, type annotations)
- 10:10-12:00: Setting up React project with TypeScript
- 12:30-2:20: TypeScript with Express (type safety for routes, middleware)
- 2:40-4:30: Convert previous projects to TypeScript
- 5:00-6:50: LeetCode Medium (dynamic programming intro) - 1-2 problems
- 7:10-8:30: Review TypeScript patterns, tsconfig setup

**Resources:**



- Official TypeScript documentation
- Maximilian Schwarzmüller "React & TypeScript: Practical Guide"
- Express TypeScript boilerplate (edwinhern/express-typescript)

### Milestones:

-  Can add types to React components
-  Express API uses TypeScript
-  Understands interfaces vs types
-  45+ LeetCode problems

**Move-on criteria:** Comfortable with basic TypeScript syntax in both frontend and backend

---

## Week 10-12: Portfolio Project #2 - Content Aggregator

**Primary objective:** Build Niche Content Aggregator with scheduled data ingestion

### Project requirements:

- Backend: Express + TypeScript + PostgreSQL
- Frontend: React + TypeScript
- Features: Content scraping/API fetching, scheduling (node-cron), search functionality
- Example niches: Tech articles, job postings, Reddit threads on specific topic

### Week 10: Backend development

- 8:00-12:00: Database schema design, API endpoints
- 12:30-4:30: Data fetching logic, scheduling implementation
- 5:00-6:50: LeetCode Medium (linked lists, two pointers)
- 7:10-8:30: Testing API endpoints, documentation

### Week 11: Frontend development

- 8:00-12:00: React components, state management
- 12:30-4:30: Connecting frontend to backend API
- 5:00-6:50: LeetCode Medium (backtracking basics)
- 7:10-8:30: UI polish, error handling


### Week 12: Integration and deployment

- 8:00-12:00: Full-stack integration, bug fixes
- 12:30-4:30: Deployment setup (backend + frontend)
- 5:00-6:50: LeetCode Medium (consolidating patterns)
- 7:10-8:30: Professional README, project documentation

### Resources:

- node-cron for scheduling
- Cheerio for web scraping (if applicable)
- Axios for API requests
- Docker basics for deployment preparation

### Milestones:

-  Backend API with scheduled jobs working

- ☒ PostgreSQL database with content storage
- ☒ React frontend consuming API
- ☒ Deployed full-stack application
- ☒ 60+ LeetCode problems (targeting 50%+ Medium success rate)

**Move-on criteria:** Can demo working full-stack application showing data collection and display

## **CHECKPOINT: End of Week 12**

- Can build full-stack PERN applications
- Solved 60+ LeetCode problems (50%+ Medium success rate)
- 2 portfolio projects deployed
- Comfortable with TypeScript

### **Red flags:**

- Project incomplete → Reduce scope (remove one feature), complete by Week 13
- LeetCode success rate below 40% Medium → Add daily 3rd problem, reduce project hours temporarily
- Can't connect frontend to backend → Spend 2 days on CORS, authentication basics before continuing

## **Phase 3: Advanced concepts and collaboration project (Weeks 13-18)**

**Learning objectives:** Master authentication, WebSockets, AWS deployment, Docker, advanced React patterns, system design basics, and complete collaborative portfolio project.

### **Week 13: Authentication and security**

**Primary objective:** Implement JWT authentication, password hashing, security best practices

#### **Daily topic rotation:**

- Topic A: Authentication concepts (JWT, sessions, OAuth)
- Topic B: bcrypt for password hashing
- Topic C: Protected routes (frontend and backend)

#### **Daily schedule:**

- 8:00-10:00: JWT concepts, token generation and validation
- 10:10-12:00: bcrypt, secure password storage
- 12:30-2:20: Middleware for protected routes
- 2:40-4:30: Build authentication system (register, login, protected endpoints)
- 5:00-6:50: LeetCode Medium (graphs, BFS/DFS patterns) - 1-2 problems
- 7:10-8:30: Security best practices review, environment variables

#### **Resources:**

- Auth0 authentication guides
- JWT.io for understanding tokens
- OWASP security guidelines (basics)

#### **Milestones:**

- ☒ Working registration and login system
- ☒ JWT tokens generated and validated
- ☒ Protected routes on both frontend and backend

-  65+ LeetCode problems

**Move-on criteria:** Can implement authentication system from scratch

---

## Week 14: WebSockets and real-time communication

**Primary objective:** Understand WebSockets, implement with Socket.io

**Daily topic rotation:**

- Topic A: WebSocket concepts vs REST
- Topic B: Socket.io setup (client and server)
- Topic C: Rooms, namespaces, broadcasting





**Daily schedule:**

- 8:00-10:00: WebSocket fundamentals, when to use vs REST [Baeldung](#) ↗
- 10:10-12:00: Socket.io server setup, connection handling
- 12:30-2:20: Socket.io client integration with React
- 2:40-4:30: Build real-time chat or notification system
- 5:00-6:50: LeetCode Medium (dynamic programming 1D) - 1-2 problems
- 7:10-8:30: Testing real-time features, debugging connections

**Resources:**

- Socket.io official documentation
- Codedamn "Real-time REST APIs: Integrating WebSockets"
- WebSocket vs REST guide (Baeldung)

**Milestones:**

-  WebSocket connection established
-  Real-time messaging between clients
-  Understands rooms/namespaces
-  70+ LeetCode problems

**Move-on criteria:** Can add real-time feature to existing application

---

## Week 15: Docker and containerization

**Primary objective:** Containerize applications, understand Docker basics

**Daily topic rotation:**

- Topic A: Docker concepts (images, containers, volumes)
- Topic B: Dockerfile creation and best practices
- Topic C: Docker Compose for multi-container apps

**Daily schedule:**





- 8:00-10:00: Docker installation, basic commands
- 10:10-12:00: Writing Dockerfile for Node.js app
- 12:30-2:20: Docker Compose (frontend + backend + database)
- 2:40-4:30: Dockerize previous projects

- 5:00-6:50: LeetCode Medium (DP continued, greedy algorithms) - 1-2 problems
- 7:10-8:30: Docker documentation, troubleshooting

### Resources:

- Docker Curriculum (docker-curriculum.com) [Docker](#) ↗
- DevOps with Docker (University of Helsinki)
- Official Docker documentation

### Milestones:

-  Created Dockerfile for full-stack app
-  Docker Compose running all services
-  Understands volumes for data persistence
-  75+ LeetCode problems

**Move-on criteria:** Can containerize full-stack application

---

## Week 16: AWS deployment basics

**Primary objective:** Deploy applications to AWS (EC2, RDS, S3, Elastic Beanstalk)

### Daily topic rotation:

- Topic A: AWS fundamentals, EC2, IAM
- Topic B: RDS (managed PostgreSQL)
- Topic C: S3 and Elastic Beanstalk deployment





### Daily schedule:

- 8:00-10:00: AWS account setup, IAM basics, EC2 concepts
- 10:10-12:00: RDS PostgreSQL setup, connecting from application
- 12:30-2:20: S3 for static file hosting
- 2:40-4:30: Deploy application to Elastic Beanstalk
- 5:00-6:50: LeetCode Medium (mixed patterns review) - 2 problems
- 7:10-8:30: AWS cost management, security best practices

### Resources:

- AWS Developer Learning Plan (official)
- AWS Free Tier documentation
- Elastic Beanstalk guides

### Milestones:

-  Application deployed to AWS
-  Database on RDS connected
-  Understands basic AWS services
-  80+ LeetCode problems (targeting 60%+ Medium success)

**Move-on criteria:** Can deploy full-stack application to cloud

---

## Week 17-18: Portfolio Project #3 - Collaborative Itinerary Planner

**Primary objective:** Build real-time collaborative application with authentication

**Project requirements:**

- **Backend:** Express + TypeScript + PostgreSQL + JWT auth + Socket.io
- **Frontend:** React + TypeScript + real-time updates
- **Features:**
  - User authentication
  - Create/edit/delete itineraries
  - Real-time collaboration (multiple users editing)
  - WebSocket updates when changes made
  - PostgreSQL for persistence






**Week 17: Core functionality**

- 8:00-12:00: Authentication system, database schema, API endpoints
- 12:30-4:30: Real-time collaboration logic, Socket.io integration
- 5:00-6:50: LeetCode Medium (targeting weak patterns) - 2 problems
- 7:10-8:30: Testing, code review

**Week 18: Polish and deployment**

- 8:00-12:00: Frontend polish, error handling, edge cases
- 12:30-4:30: Deployment (Docker + AWS or Heroku + PostgreSQL)
- 5:00-6:50: LeetCode Medium (interview prep focus) - 2 problems
- 7:10-8:30: Documentation, README, prepare for showcase

**Milestones:**

-  Full authentication system
-  Real-time collaboration working
-  Deployed with database persistence
-  Professional documentation
-  90+ LeetCode problems (65%+ Medium success rate)

**Move-on criteria:** Project is **complete, deployed, and demo-ready**

**CHECKPOINT: End of Week 18**

- Strong full-stack skills with modern technologies
- Solved 90+ LeetCode problems (65%+ Medium success rate)
- 3 portfolio projects demonstrating progression
- Ready to start job applications

**Red flags:**

- Project incomplete → Extend to Week 19, reduce scope if needed
- LeetCode below 60% Medium → Dedicate more time to algorithms in final weeks
- Can't explain technical decisions → Review architecture, add comments

---

**Phase 4: Job preparation and advanced topics (Weeks 19-20)**

**Learning objectives:** Interview preparation, system design basics, portfolio polish, begin job applications, start Project #4 foundation.

## Week 19: Interview preparation intensive

**Primary objective:** Behavioral interview prep, portfolio polish, application materials

**Daily schedule** (adjusted focus):

- 8:00-10:00: Behavioral interview prep (STAR stories, practice)
- 10:10-12:00: Mock interviews (schedule with mentor)
- 12:30-2:20: Portfolio polish (READMEs, deployment checks, GitHub cleanup)
- 2:40-4:30: Resume/cover letter writing, LinkedIn optimization
- 5:00-6:50: LeetCode Medium/Hard mix - 2 problems
- 7:10-8:30: Company research, start applications (5-10 companies)







**Activities:**

- **Monday:** Write 5 STAR stories (challenges, learning, collaboration, failure, technical problem) [mit](#) ↗
- **Tuesday:** Mock interview with mentor (behavioral + technical)
- **Wednesday:** Polish all 3 portfolio projects, update READMEs [Web Portfolios](#) ↗ [Indeed](#) ↗
- **Thursday:** Resume/LinkedIn optimization, GitHub profile README [finalroundai](#) ↗
- **Friday:** Research 20 target companies, begin applications

**Resources:**

- MIT STAR Method Worksheet
- Pramp for peer mock interviews [Pramp](#) ↗ [Design Gurus](#) ↗
- interviewing.io for practice [Interviewing](#) ↗ [Design Gurus](#) ↗
- Canadian job boards (Indeed.ca, LinkedIn)

**Milestones:**

-  5 polished STAR stories
-  Mock interview completed with feedback
-  All projects have professional READMEs
-  Resume tailored for Canadian tech market
-  Applied to 10+ positions
-  100+ LeetCode problems (70%+ Medium success rate)

**Move-on criteria:** Application materials ready, first batch of applications sent

---

## Week 20: System design and Project #4 kickoff

**Primary objective:** Learn system design basics, start Auction Marketplace project

**Daily schedule:**

- 8:00-10:00: System design fundamentals (load balancing, caching, databases)
- 10:10-12:00: Study junior-level system design questions
- 12:30-4:30: Begin Project #4 planning and initial setup
- 5:00-6:50: LeetCode contest participation or hard problems
- 7:10-8:30: Continue applications (daily 5-10), networking

**System design topics** (junior level):

- Client-server architecture
- REST API design

- SQL vs NoSQL decisions
- Caching basics (Redis concepts)
- Load balancer purpose
- Basic scaling concepts

#### Project #4: Auction Marketplace (start during job search)

- **Scope:** Continue building while applying to jobs
- **Features:** Stripe payments, AWS S3 for images, bidding system
- **Timeline:** Complete over Weeks 20-24+ during job search

#### Resources:

- IGotAnOffer system design guide (junior level) [igotanoffer](#) ↗
- System Design Primer (GitHub)
- Grokking the System Design Interview (Educative)

#### Milestones:

- ☒ Can discuss basic system architecture
- ☒ Project #4 planning completed
- ☒ Applied to 30+ positions total
- ☒ 110+ LeetCode problems (70%+ Medium success)

#### FINAL CHECKPOINT: End of Week 20

- **Technical skills:** Can build production-ready full-stack applications
- **Algorithms:** 110+ LeetCode problems, 70%+ Medium success rate
- **Portfolio:** 3 complete projects deployed, 4th project started
- **Job search:** 30+ applications, active networking
- **Interview ready:** Behavioral stories prepared, technical skills strong

**Transition to Month 5+:** Continue Project #4 while in active job search (applying 5-10 daily, networking, interviews)

---

## Detailed resource recommendations

### Frontend (React + TypeScript)

#### Primary course (choose one):

- **Jonas Schmedtmann - Ultimate React Course 2025** (Udemy, \$15-20): Project-heavy, 68 hours, covers React 19 + TypeScript
- **Full Stack Open** (Free): University of Helsinki, exercise-driven, modern stack, 200+ hours with optional modules

#### TypeScript:

- **Maximillian Schwarzmüller - React & TypeScript** (Udemy, \$15-20): Practical patterns, 15 hours
- Official TypeScript documentation (reference)

#### Supplementary:

- Official React docs (react.dev) - excellent for reference
- Frontend Masters - Complete Intro to React v9 (Brian Holt) if budget allows (\$39/month)

# Backend (Node.js + Express + TypeScript)

## Primary resources:

- **Auth0 - Node.js and TypeScript Tutorial:** Free, CRUD API, production patterns
- **Express TypeScript Boilerplate** (edwinhern/express-typescript GitHub): Study production code
- Official Express and Node.js documentation

## Supplementary:

- LogRocket "Express with TypeScript 2024" tutorial
- Toptal "Express.js REST API with TypeScript"

# Database (PostgreSQL)

## Primary:

- **PostgreSQLTutorial.com:** Free, comprehensive, practical examples [Neon](#)
- Official PostgreSQL documentation (reference)

## Supplementary:

- SQL and PostgreSQL for Beginners (Udemy, \$15-20) if you prefer video
- node-postgres library documentation

# Docker

## Primary:

- **Docker Curriculum** (docker-curriculum.com): Free, 4-6 hours, practical [Docker](#)
- **DevOps with Docker** (University of Helsinki): Free, 30-40 hours with exercises

## Supplementary:

- Docker & Kubernetes Practical Guide 2024 (Udemy, \$15-20) for deeper understanding

# AWS

## Primary:

- **AWS Developer Learning Plan** (official): Free digital courses, 30-40 hours
- AWS Free Tier documentation

## Supplementary:

- Full Stack Apps on AWS (Udacity) if budget allows

# Git/GitHub

## Primary:

- Official Git tutorial
- MIT Missing Semester - Version Control lecture (free)
- GitHub documentation for advanced workflows



**Supplementary:**

- Mastering Git & GitHub (Udemy, \$15-20) for comprehensive coverage

**Other essentials**

**REST APIs + WebSockets:**

- Socket.io official documentation
- Auth0 REST API tutorials
- Baeldung WebSocket vs REST guide

**Stripe:**

- Official Stripe documentation (best-in-class)
- DEV.to "Stripe Subscription Integration Ultimate Guide 2024"
- Stripe CLI for testing

**Best free curriculum option: Full Stack Open** covers React, Node.js, Express, testing, TypeScript, and more in one structured path. Highly recommended as primary resource.

**Budget recommendation:**

- **\$0-50/month:** Totally viable with free resources
  - **\$50-100/month:** Add Udemy courses (\$60 total) + Frontend Masters (\$39/month)
  - **Investment mindset:** Quality courses pay for themselves
- 

**LeetCode study plan integration**

**Weekly problem targets**

- Weeks 1-4:** 5 Easy per week (20 total)
- Weeks 5-8:** 3 Easy + 5 Medium per week (60 cumulative)
- Weeks 9-12:** 2 Easy + 6 Medium per week (90 cumulative)
- Weeks 13-16:** 1 Easy + 7 Medium per week (120 cumulative)
- Weeks 17-20:** 8 Medium per week (140+ cumulative target)

**Pattern-based progression (NeetCode 150 recommended)**

**Phase 1 patterns (Weeks 1-6):**

- Arrays & Hashing (12 problems)
- Two Pointers (5 problems)
- Sliding Window (6 problems)
- Stack (7 problems)
- Binary Search (7 problems)

**Phase 2 patterns (Weeks 7-12):**

- Linked Lists (6 problems)
- Trees (15 problems)
- Tries (3 problems)
- Heap/Priority Queue (7 problems)
- Backtracking basics (9 problems)

**Phase 3 patterns** (Weeks 13-18):

- Graphs (8 problems)
- 1-D Dynamic Programming (12 problems)
- 2-D Dynamic Programming (11 problems)
- Greedy (8 problems)

**Phase 4 patterns** (Weeks 19-20):

- Intervals (6 problems)
- Math & Geometry (8 problems)
- Mixed hard problems

**Daily LeetCode schedule (BLOCK 5: 5:00-6:50 PM)**

**Standard approach** (110 minutes):

- **5:00-5:45:** Problem 1 attempt (45 min maximum)
- **5:45-6:00:** Review solution if stuck, understand approach
- **6:00-6:45:** Problem 2 attempt or implement alternative solution
- **6:45-6:50:** Log problems in tracker, note patterns

**Spaced repetition:**

- Day 3: Re-solve problems from current week
- Day 7: Re-solve problems from last week
- Day 14: Review pattern from 2 weeks ago

**Critical rules**

**Time-boxing:** 30-45 minutes per Medium problem max, then look at solution  
**No AI for algorithms:** Zero tolerance. Build pattern recognition independently.  
**Understand before moving on:** Can you solve it again tomorrow?  
**Track patterns:** Maintain spreadsheet with problem, pattern, difficulty rating, dates solved

**Success metrics**

- **Week 6:** 80%+ Easy, 40%+ Medium
- **Week 12:** 90%+ Easy, 50%+ Medium
- **Week 18:** 95%+ Easy, 65%+ Medium
- **Week 20:** 70%+ Medium (consistent), can solve most in 25-30 minutes

**Resources**

- **NeetCode.io:** Free roadmap with video explanations
- **Grind 75:** Customizable study plan (<https://www.techinterviewhandbook.org/grind75/>)
- **Patterns:** Create personal cheat sheet for each pattern

---

**AI tool usage framework**

**Tool selection by task**

**Perplexity Pro:**

- Researching technical concepts with citations
- "What are best practices for JWT authentication in Node.js?" (gets cited sources)
- Comparing different approaches to same problem
- Learning new technologies (get overview with sources)

### ChatGPT Pro:

- Quick explanations of confusing concepts
- Generating practice problems (not solutions)
- Brainstorming project ideas
- General programming questions

### Claude Max:

- Code review of YOUR completed code
- Debugging assistance (after you've tried 30 minutes)
- Large codebase understanding
- Detailed explanations of complex patterns

### Google Gemini Pro:



- Document analysis (if working with technical specs)
- Multi-modal tasks
- Integration with Google ecosystem

### GitHub Copilot:



- **AVOID during learning phase** (Weeks 1-16)
- Only use Weeks 17-20 for boilerplate, NOT logic
- Disable when doing LeetCode (always)

## The learning hierarchy (NO AI zones)



### Phase 1 (Weeks 1-6): 90% manual

-  NO AI for: Writing any code solutions, debugging basic errors, algorithm practice
-  AI allowed for: Clarifying confusing terms, alternative explanations



### Phase 2 (Weeks 7-12): 70% manual

-  NO AI for: Core logic, algorithm practice, debugging (try 30 min first), project architecture
-  AI allowed for: Boilerplate setup, explaining unfamiliar patterns you encounter, reviewing YOUR code

### Phase 3 (Weeks 13-18): 50% manual

-  NO AI for: Algorithm practice (always), core features, architecture decisions
-  AI allowed for: Boilerplate, library exploration, code review, refactoring suggestions

### Phase 4 (Weeks 19-20): Strategic use

-  NO AI for: Interview practice, algorithm practice, explaining your projects
-  AI allowed for: Mock interview feedback, system design discussion, documentation

## Prompting as tutor (not code generator)

### GOOD prompts (encourages learning):



"Explain how JWT authentication works conceptually. Don't give me code yet, help me understand the flow first. Then ask me questions to check my understanding."

"I wrote this authentication function [paste]. Can you explain what's wrong with my approach and why? Don't fix it for me."

"Generate 5 practice problems about recursion with hints, not solutions."

**BAD prompts** (creates dependency):



"Write a JWT authentication system for me"

"Fix this bug [paste code]"

"Build a REST API for my project"

**The AI debugging protocol**

- Step 1:** Try yourself for 30 minutes (15 min for beginners)
- Step 2:** Google the exact error message
- Step 3:** Ask AI for hints, not solutions: "Why might this error occur? What should I investigate?"
- Step 4:** Only after understanding the problem, ask: "Can you show me the fix and explain why?"

**Weekly AI usage limits**

- **Weeks 1-6:** Max 3 code generations per day (for setup/boilerplate only)
- **Weeks 7-12:** Max 5 AI assists per day
- **Weeks 13+:** Unlimited but track dependency

**Dependency test** (take monthly):

- Can I build a simple CRUD app without AI? (Should be YES by Week 8)
- Can I debug basic errors without AI? (Should be YES by Week 6)
- Can I explain all code in my projects? (Should be ALWAYS YES)

If answering NO → **Reduce AI use by 50% immediately**

**AI for code review**

**After YOU complete code:**



"Review this code for a [your level] developer:

[PASTE CODE]

Analyze:

1. Security vulnerabilities
2. Performance issues
3. Best practices violations
4. Potential edge cases

For each issue, explain WHY it's a problem and let me propose a fix first."

**Then:** Implement changes one at a time, understanding each

## Red flags of AI dependency

- Can't start coding without AI prompts
- Panic when AI unavailable
- Can't explain code you "wrote" yesterday
- Declining problem-solving ability
- Every project uses AI-generated boilerplate you don't understand

**If experiencing any** → Take 1-2 weeks with ZERO AI use to rebuild fundamentals

---

## Strategic mentor usage

### Weekly mentor sessions (recommended schedule)

**Week 1-4:** 30-minute check-ins

- Code review of practice exercises
- Clarify confusing concepts
- Adjust learning pace if needed

**Week 5-6:** 1-hour project review

- Architecture review of Pathfinding Visualizer
- Code quality feedback
- Discuss challenges overcome

**Week 7-12:** 45-minute weekly reviews

- Backend code review
- Database schema design feedback
- API design patterns

**Week 13-18:** 1-hour sessions

- Full-stack project review

- Architecture discussions
- Real-time features review

## **Week 19-20:** Mock interviews

- 2-3 full mock interviews (behavioral + technical)
- Resume review
- Interview feedback

## **Pre-session preparation**

### **24 hours before:**

- Push code to GitHub
- Write specific questions list (3-5 questions)
- Document what you tried before asking
- Prepare demo of current project state

### **Example questions:**

- "I implemented authentication this way [explain]. Is this secure? What would you change?"
- "I'm stuck on [specific problem]. I tried [X, Y, Z]. What am I missing?"
- "Can you review my database schema for [project]?"

## **During sessions**

- **Show, don't just tell:** Demo your working code
- **Ask specific questions:** Not "How do I build this?" but "I tried this approach, why didn't it work?"
- **Take notes:** Document feedback immediately
- **Challenge yourself:** Ask mentor to give you a problem to solve on the spot

## **Post-session**

- Review notes within 24 hours
- Implement top 3 feedback items
- Document learnings in project README

## **Mock interview schedule**

**Week 8:** First mock interview (basic algorithms)

**Week 14:** Second mock interview (full-stack questions)

**Week 19:** Intensive mock interviews (2-3 sessions)

**Week 20:** Final polish interview

### **Mock interview structure:**

- 10 min: Behavioral question (STAR method)
  - 30 min: Technical coding problem
  - 15 min: System design or project discussion
  - 5 min: Questions for interviewer
-

# Mental health and burnout prevention

## Non-negotiable structure (for depression management)

**Daily anchors** (do regardless of motivation):

1. **Wake same time** (6:00 AM, even weekends initially)
2. **Morning movement** (20 min walk/yoga - not optional)
3. **Breakfast** (pre-decided meal, no choice paralysis)
4. **Start studying by 8:00 AM** (momentum builds from action, not feeling)
5. **Lunch at 12:00 PM** (set alarm if needed)
6. **Mandatory breaks** (not based on feeling productive)
7. **Stop by 8:30 PM** (protect evening wind-down)
8. **Sleep by 10:30 PM** (7.5-8 hour minimum)

## Decision elimination systems

**Pre-decide:**

- Same breakfast every weekday (one less decision)
- Study location set up night before
- Tomorrow's schedule written tonight (follow, don't create)
- Meal prep Sundays (eliminates daily food decisions)

**Automation:**

- Calendar blocks for all study sessions (follow blindly)
- Focusmate scheduled in advance (external accountability)
- Break reminders automated (can't skip)
- Weekly review template (checklist, no thinking)

## Burnout warning signs

**STOP and take 2-3 days off immediately if:**

- Physical symptoms: chronic headaches, insomnia, frequent illness
- Emotional: hopelessness about progress, crying, irritability spikes
- Cognitive: can't concentrate worse than baseline, memory issues
- Behavioral: skipping self-care (showers, meals), social withdrawal beyond normal

**Reduce to 50% intensity if:**

- Motivation declining multiple days straight
- Taking longer to code than previous weeks
- Avoiding starting projects
- Increase in negative self-talk

## Weekly sustainability check (Sunday evening)

Rate 1-10:

- Energy level this week: \_\_\_\_
- Mood this week: \_\_\_\_
- Sleep quality: \_\_\_\_
- Exercise days: \_\_\_\_

- Social connection: \_\_\_\_

**If any below 4** → Adjust next week's schedule (use reduced template)

**If multiple below 6** → Consider extending timeline

## **The 3-month danger zone**

**Weeks 12-16 are highest burnout risk** (novelty gone, job not yet in sight)

**Preventive measures:**

- Extra rest days in Weeks 12, 16
- Connect with other learners (bootcamp Discord, study groups)
- Celebrate milestones (project completions)
- Vary routine slightly (different study location, time of day)
- Remember your "why" (write it down, reread monthly)

## **When to adjust timeline**

**Add 2-4 weeks if:**

- Consistent burnout symptoms
- Depression worsens significantly
- Missing milestones by more than 1 week repeatedly
- Physical health deteriorating

**Better to take 6 months with health intact than burnout in 4 months**

## **Emergency mental health resources**

- BetterHelp/Talkspace: Online therapy
- 7 Cups: Free peer support
- Crisis Text Line: Text "HELLO" to 686868 (Canada)
- Your existing mental health provider

**Remember:** Career success means nothing without health. Protect yourself first.

---

## **Job application timeline**

### **When to start applying**

**Start applications at Week 19** (after 3 portfolio projects complete)

**DON'T wait for:**

- Feeling "ready" (you won't)
- Perfect portfolio (doesn't exist)
- Mastering everything (impossible)
- 100% confidence (no one has this)



# Application strategy

**Weeks 19-20:** 10-15 applications per week  
**Month 5+** (during Project #4): 25-30 applications per week  
**Ramp to:** 5-10 applications per day during active search

## Target mix:

- 40% - Junior/Entry-level positions
- 30% - Mid-level (if 60%+ qualified, apply anyway)
- 20% - Startups (more open to career changers)
- 10% - Agencies (value diverse backgrounds)

## Canadian companies to target

### Major tech:

- Shopify (Ottawa/Toronto)
- Wealthsimple (Toronto)
- Hootsuite (Vancouver)
- 1Password (Toronto)

### Startups:

- ApplyBoard (Waterloo)
- Clearco (Toronto)
- TouchBistro (Toronto)
- Faire (Toronto)

### Job boards:

- Indeed.ca
- LinkedIn Jobs (set to Canada)
- AngelList (startups)
- VanHack (Canadian tech + remote)
- BetaKit (Canadian startup news + jobs)

## Behavioral interview prep

### Create 5 STAR stories (Week 19):

1. **Technical challenge solved:** Debugging complex issue, implementing difficult feature
2. **Learning something new quickly:** Career transition, learning new technology
3. **Team collaboration:** Group project, helping peer, code review
4. **Failure and learning:** Project that didn't work, mistake made, what you learned
5. **Initiative/leadership:** Starting project, suggesting improvement, teaching others

### STAR template:

- **Situation** (20%): 2-3 sentences of context
- **Task** (10%): Your specific responsibility
- **Action** (60%): YOUR actions (use "I" not "we"), specific steps
- **Result** (10%): Quantified outcome, what you learned

# System design for junior roles

**Reality:** Most junior roles DON'T ask system design

**If asked, expectations:**

- Basic client-server-database architecture
- Can explain REST API design
- Understands when to use SQL vs NoSQL (high level)
- Knows what caching and load balancers do
- NOT expected: Production-ready designs, scaling to millions of users

**Preparation** (Week 20):




- IGotAnOffer junior system design guide
- Practice explaining your portfolio project architectures
- Understand trade-offs you made

## Realistic timeline




- **Applications sent:** 100-200 before offer (normal)
- **Response rate:** 5-15%
- **Time to offer:** 3-6 months active searching
- **Rejection rate:** 85-95% (this is NORMAL)

## Success mindset

**Reframe:**

-  "I bring unique perspective from [previous field]"
-  "Each interview teaches me what to improve"
-  "Companies need diverse backgrounds"

**NOT:**




-  "I'm just a bootcamp grad"
-  "Everyone has more experience"
-  "I'll never be good enough"

---



## Red flags and contingency plans

### Week-by-week red flags

**Weeks 1-6:**




-  Less than 20 LeetCode solved → Add 3rd daily problem
-  Can't build React app without tutorial → Redo Week 4-5 with different tutorial
-  Project #1 incomplete → Reduce scope, finish in Week 7

**Weeks 7-12:**




-  Can't connect frontend to backend → 2-day focused tutorial on CORS, REST
-  Less than 50 LeetCode solved → Increase daily practice to 3 problems

-  Project #2 incomplete → Extend to Week 13, cut one feature

### **Weeks 13-18:**




-  Medium success rate below 50% → Add 1 hour daily algorithm time
-  Can't implement authentication → Follow step-by-step tutorial once, then rebuild
-  Project #3 incomplete → Extend to Week 19, simplify real-time features

### **Weeks 19-20:**




-  Can't explain projects in interviews → Review code, add comments, write explanations
-  No callbacks from applications → Review resume, tailor more, increase networking
-  Failing technical interviews → More mock interviews, identify weak patterns

## **General red flags**




### **Learning pace issues:**

-  Consistently behind by 1+ weeks
-  Forgetting material from 2 weeks ago
-  Can't complete projects without AI/tutorials

### **Mental health concerns:**

-  Burnout symptoms (see section above)
-  Depression worsening
-  Missing multiple days of study

### **Skill gaps:**

-  Algorithm success rate not improving
-  Can't debug independently
-  Interview performance not improving

## **Contingency plans**

### **If 1-2 weeks behind:**

- Use Friday reduced schedule to catch up
- Skip one LeetCode session to focus on projects
- Extend current phase by 1 week

### **If 3+ weeks behind:**

- Reassess entire timeline, add 4-6 weeks
- Identify bottleneck: concepts vs practice vs mental health
- Consider focusing on 2 portfolio projects (not 3) to save time
- Reduce LeetCode target to 80-90 problems (60% Medium success acceptable)

### **If burnout occurs:**

- STOP immediately for 2-3 days (non-negotiable)
- Return at 50% intensity for 1 week
- Extend timeline by 2-4 weeks
- Add more rest days permanently

**If not getting interviews by Week 22-24:**

- Resume/application review with mentor
- Increase networking (coffee chats, meetups)
- Tailor applications more (quality over quantity briefly)
- Consider contract/freelance work to build experience

**Flexibility principles**

**Rigid on:** Daily structure, breaks, sleep, non-negotiables

**Flexible on:** Specific topics, project scope, timeline length

**When in doubt:** Ask yourself:

1. Am I learning and making progress (even if slow)?
2. Is my mental health okay (or getting help)?
3. Am I moving toward the goal?

If YES to all three → Keep going, adjust pace as needed





If NO to any → Stop, reassess, get support

---



**Success metrics and milestones**

**Monthly checkpoints**





**End of Month 1 (Week 4):**

-  20+ LeetCode problems (80%+ Easy success)
-  Can build interactive webpage with vanilla JS
-  Comfortable with Git basics
-  Daily study routine established





**End of Month 2 (Week 8):**

-  40+ LeetCode problems (40%+ Medium success)
-  1 portfolio project deployed (Pathfinding Visualizer)
-  Can build React applications independently
-  Understands full-stack architecture





**End of Month 3 (Week 12):**

-  60+ LeetCode problems (50%+ Medium success)
-  2 portfolio projects deployed
-  Can build complete PERN stack applications
-  Comfortable with TypeScript

**End of Month 4 (Week 16):**

-  80+ LeetCode problems (60%+ Medium success)
-  Understands authentication, real-time features, Docker, AWS
-  Can deploy applications to cloud
-  Strong full-stack foundation

**End of Month 5 (Week 20):**

-  110+ LeetCode problems (70%+ Medium success)
-  3 portfolio projects complete, 4th started
-  Applied to 30+ jobs
-  Interview-ready (technical and behavioral)

## Technical skill validation

### Can you:

- Build a CRUD app from scratch in 4-6 hours? (Should be YES by Week 12)
- Explain your technical decisions clearly? (Always YES)
- Debug independently for 30 minutes before asking for help? (YES by Week 8)
- Read others' code and understand it? (YES by Week 16)
- Complete Medium LeetCode in 25-30 minutes? (YES for 70%+ by Week 20)

## Portfolio quality check

### Each project should have:

- Live deployed demo
- Professional README (problem, solution, tech stack, challenges, features)
- Clean, commented code
- Git history showing iteration
- Screenshots/GIFs
- Can demo in 5-10 minutes

## Interview readiness check

### Behavioral:

- 5 STAR stories prepared and practiced
- Can explain career change narrative confidently
- Researched target companies

### Technical:

- Can solve 70%+ of Medium LeetCode problems
- Can explain portfolio projects in detail
- Can discuss trade-offs and alternative approaches
- Comfortable with mock interview pressure

### Confidence markers:

- You call yourself a "developer" (not "aspiring")
- You can discuss technologies naturally
- You're excited (not terrified) to talk about projects
- You've helped others with code questions

## Long-term success definition

### Month 6-9 (realistic job offer timeline):

- Continued learning during job search
- 100-200 applications sent
- Multiple interview rounds

- Job offer(s) received
- Successful career transition complete

**Remember:** Landing first developer job is the goal, not becoming senior engineer. You'll learn most on the job. This curriculum gets you interview-ready and job-ready for junior positions in the Canadian market.

---

## Final notes

### This curriculum is aggressive

10-14 hours daily study for 4-5 months is at the upper limit of human capacity. **Not everyone can or should do this.** This is designed for someone with:

- Full-time availability
- Strong internal motivation
- Support system (mentor, family)
- Ability to manage depression with structure








### Permission to adjust

- Take 6 months instead of 4-5 if needed
- Reduce to 8 hours daily if 10-14 is unsustainable
- Skip Project #3 if timeline is tight (2 excellent projects > 3 rushed)
- Hit 80-90 LeetCode instead of 110+ if that's your limit

**Success = employed as developer with skills intact and health maintained**

### You've got this

You have:

-  Strong theoretical background (number theory paper)
-  Proven ability to learn (C, Python, Racket, LeetCode)
-  Built projects before (Snake game)
-  Support system (senior dev mentor)
-  Time commitment (10-14 hours daily)
-  Awareness of mental health needs
-  Access to top AI tools

This curriculum **eliminates decisions** while giving you **structure to execute**. Your job is to follow the plan, adjust as needed, take care of yourself, and keep moving forward.

**One day at a time. One block at a time. One project at a time.**

You're not "aspiring" to be a developer. From Week 1, **you are a developer who is building skills**. Own that identity.

### Emergency contact

If you're struggling (mental health, technical, motivation):

- Reach out to your mentor
- Join supportive communities (freeCodeCamp Discord, The Odin Project)
- Take a break if needed (better to pause than burnout)

- Remember: This is a marathon, not a sprint

**Good luck. You've got this. See you on the other side as a professional developer.**