predev. Academy

TRACK ROADMAP

React and Next.js Web Development

Build and deploy a production web app with login, a dashboard, Arabic and English, and tests, using React 19 and Next.js 15.

Web Frontend

DURATION	SESSIONS	SCHEDULE	FORMAT	LEVEL	LANGUAGE
10 weeks	20 live sessions, 60 hours	Twice a week, 3 hours per session, plus 1 hour of online office hours every week	In person or online, live	Beginner, after Tech Compass	Egyptian Arabic, English technical terms

Who it is for

- CS and IT students who want a real frontend skill set before graduation
- Fresh graduates who know some programming and want their first web job
- Career switchers who finished Tech Compass and chose web development
- Designers or WordPress builders who want to write modern React code

Before day 1

- Completed the Tech Compass intro, or can show basic programming (variables, conditions, loops, functions)
- Laptop with 8 GB RAM minimum (16 GB recommended), SSD, Windows 10/11, macOS, or Linux
- Installed before day 1: Node.js 24 LTS, Git, VS Code, Google Chrome
- Free accounts: GitHub, Vercel, and a Google account for shared docs
- Comfortable reading simple English documentation (we explain in Arabic)

By the end you can

- Build accessible, responsive pages with semantic HTML and modern CSS that work in Arabic RTL and English
- Write clean modern JavaScript and TypeScript, including async code and API calls with fetch
- Build React 19 interfaces with components, props, state, effects, and custom hooks
- Structure a Next.js 15 App Router app with layouts, nested routes, server and client components, and server actions
- Build validated forms with React Hook Form and Zod, and manage server data with TanStack Query
- Add login, sessions, and role based protected pages to a web app
- Ship a bilingual Arabic and English interface with correct RTL layout
- Write unit, component, and end-to-end tests, and use them as a safety net when coding with AI
- Measure and improve Core Web Vitals and add SEO metadata
- Deploy a production app to Vercel or a VPS with Coolify, and present it in a GitHub portfolio

Tech stack

WEB BASICS

HTML5	Semantic structure that search engines and screen readers understand
CSS (Flexbox, Grid, logical properties)	Responsive layouts that work in Arabic RTL and English LTR
JavaScript (ES2024)	The language of the browser and of every React app
TypeScript 5	Catches mistakes before runtime and is expected in most job posts

FRAMEWORK AND UI

React 19	The most requested frontend library in Egypt and in remote jobs
Next.js 15 App Router	Routing, server rendering, data fetching, and server actions in one framework
Tailwind CSS 4	Fast, consistent styling that teams use daily
shadcn/ui	Accessible components you own and can change, built on Radix

DATA, FORMS, AND AUTH

TanStack Query 5	Caching, loading states, and mutations for client side data
React Hook Form + Zod	Fast forms with one validation schema shared by client and server
Auth.js	Login, sessions, and protected routes without writing auth from zero
next-intl	Arabic and English routes, messages, and number and date formats

QUALITY AND DELIVERY

Vitest + Testing Library	Unit and component tests that run in seconds
Playwright	One end-to-end test of the main user flow in a real browser
Git and GitHub	Version control, pull requests, and code review like a real team
Vercel or Coolify on a VPS	A real public deploy with environment variables and a custom domain
Claude Code or Cursor	AI assisted coding with review, tests, and clear limits

The running project: Mawid: clinic booking web app with admin dashboard

Mawid is a booking platform for a small clinic chain: patients pick a branch, doctor, and time slot, and staff manage everything from an admin dashboard. It mirrors the booking and admin dashboards predev. Solutions builds for clients, so students work on the same problems a real team faces: auth, roles, forms, tables, filters, and Arabic and English users.

- Public landing page with branches, doctors, and services, with SEO metadata
- Patient sign up and login with protected routes
- Booking flow: choose branch, doctor, date, and an available time slot
- My bookings page with cancel and reschedule
- Admin dashboard with stats cards and a simple chart
- Bookings table with search, filters, sorting, and pagination
- Doctors and services management with create, edit, and delete forms
- Role based access for admin, receptionist, and patient
- Full Arabic and English support with RTL layout and a language switcher
- Light and dark theme, loading and error states, toasts
- Unit and component tests plus one Playwright test for the booking flow
- Production deploy with environment variables and a custom domain

Phase overview

WEEKS 1-3

Foundations

Write semantic HTML, responsive CSS, and modern JavaScript and TypeScript, and use Git and GitHub every day.

WEEKS 4-7

Core framework

Build the Mawid booking app step by step with React 19, Next.js 15 App Router, Tailwind, forms, data fetching, auth, Arabic and English, and tests.

WEEKS 8

Building with AI

Use AI coding tools to work faster while keeping control through review, tests, and clear limits.

WEEKS 9

Capstone

Build your own web app from a brief and deploy it to production.

WEEKS 10

Career week

Get ready to apply: CV, LinkedIn, GitHub portfolio, interviews, freelancing, and demo day.

Week by week

Foundations · Weeks 1-3

Week 1 How the web works, HTML, and Git

Session 1 The web, the browser, and semantic HTML

- What happens when you open a URL: DNS, HTTP request, response, rendering
- Browser DevTools: Elements, Console, Network
- HTML document structure, head and meta tags, lang and dir attributes
- Semantic elements: header, nav, main, section, article, footer
- Links, images with alt text, lists, and tables
- Forms: input types, labels, buttons, and built in validation

In-class lab: Build a semantic multi section page for a clinic: header, services list, doctors table, and a contact form, checked with the DevTools accessibility tree.

Session 2 Git, GitHub, and problem solving habits

- Terminal basics: folders, files, and running commands
- Git: init, add, commit, log, diff, and good commit messages
- Branches, merge, and fixing a simple merge conflict
- GitHub: repos, push, pull, README files, and GitHub Pages
- Problem solving: read the problem, write steps in plain words, then code
- How to search docs and read error messages

In-class lab: Put the clinic page in a GitHub repo using a feature branch and a pull request, then publish it with GitHub Pages.

Homework: Build a personal profile page in semantic HTML, push it to GitHub with at least 5 clear commits, and publish it on GitHub Pages.

Week 2 CSS layout, responsive design, and RTL

Session 1 CSS fundamentals and Flexbox

- Selectors, specificity, and the cascade
- Box model, margin, padding, border, and box-sizing
- Colors, units (rem, %, vw), and typography with web fonts
- Flexbox: direction, alignment, wrapping, and gap
- CSS variables for colors and spacing

In-class lab: Style the clinic page: navbar and doctor cards built with Flexbox, with a color and spacing system in CSS variables.

Session 2 Grid, responsive layouts, and Arabic RTL

- CSS Grid: columns, rows, areas, and auto-fit with minmax
- Mobile first design and media queries
- Logical properties: margin-inline, padding-block, inset-inline-start
- dir="rtl", Arabic fonts, and mirroring icons correctly
- Basic accessibility: color contrast, focus styles, and visible labels

In-class lab: Turn the clinic page into a responsive grid that works from 360px to desktop, then switch it to Arabic RTL without breaking the layout.

Homework: Rebuild a real Egyptian service landing page (pick one) from a screenshot, responsive and in both RTL and LTR, and deploy it on GitHub Pages.

Week 3 Modern JavaScript and TypeScript basics

Session 1 JavaScript you will use every day

- let and const, types, template strings, and truthy and falsy values
- Functions, arrow functions, and scope
- Arrays and objects: map, filter, reduce, find, spread, and destructuring
- Modules: import and export
- The DOM: selecting elements, events, and updating the page
- Small problem solving drills with arrays and strings

In-class lab: Build a doctor search page in plain JavaScript: filter a list of doctors by name and specialty as the user types.

Session 2 Async JavaScript, fetch, and TypeScript

- The event loop in simple terms, Promises, async and await
- fetch, JSON, HTTP methods, and status codes
- Error handling with try and catch, loading and empty states
- TypeScript: why it exists, basic types, type and interface
- Union types, optional fields, and typing function parameters and returns
- Running TypeScript in a Vite project

In-class lab: Fetch doctors from a real JSON API and render them with loading and error states, then convert the code to TypeScript with typed API responses.

Homework: Solve 6 short problem solving tasks in TypeScript (arrays, strings, objects), and build a small typed app that calls a public API and shows results with filters.

Core framework · Weeks 4-7

Week 4 React 19 fundamentals

Session 1 Components, props, and JSX

- Why React: components and declarative UI
- Creating a React 19 app with Vite and TypeScript
- JSX rules, rendering lists with keys, and conditional rendering
- Props and typing props with TypeScript
- Splitting a page into small reusable components
- React DevTools

In-class lab: Build the Mawid doctors page as components: DoctorCard, DoctorList, and SpecialtyBadge, with typed props and mock data.

Session 2 State, events, effects, and hooks

- useState and handling events
- Lifting state up and passing callbacks
- useEffect: when to use it and when not to
- Controlled inputs and simple forms
- useRef, useContext, and writing a custom hook
- Rules of hooks and common mistakes

In-class lab: Add search, specialty filter, and a favorite doctors list stored in context, and extract the filter logic into a useDoctorFilter custom hook.

Homework: Build a time slot picker component that shows available slots for a selected day and highlights the chosen slot, with typed props and clean component structure.

Project milestone: Mawid doctors list and slot picker work in React with mock data.

Week 5 Next.js 15 App Router, Tailwind 4, and shadcn/ui

Session 1 Routing, layouts, and styling

- Creating a Next.js 15 project and the app folder structure
- File based routing, dynamic routes, and route groups
- Layouts, loading.tsx, error.tsx, and not-found.tsx
- Tailwind CSS 4: utility classes, theme tokens, and responsive variants
- shadcn/ui: adding Button, Card, Dialog, Table, and Form components
- Light and dark theme

In-class lab: Move Mawid to Next.js: public layout, dashboard layout with a sidebar, doctors and doctor details pages, styled with Tailwind and shadcn/ui.

Session 2 Server and client components and data fetching

- Server components versus client components and the use client directive
- Fetching data in server components with async and await
- Caching basics: static and dynamic rendering, revalidate
- Route handlers for a small API
- Streaming with Suspense and skeleton loaders
- Environment variables in Next.js

In-class lab: Fetch branches, doctors, and services from the Mawid API in server components, add skeleton loading states, and handle API errors with error.tsx.

Homework: Build the services page and the doctor details page with a dynamic route, metadata, loading and error states, and push it as a reviewed pull request.

Project milestone: Mawid runs on Next.js with public pages and the dashboard shell, reading real data from the API.

Week 6 Forms, mutations, and client data

Session 1 Forms with React Hook Form, Zod, and server actions

- React Hook Form: register, errors, and submit state
- Zod schemas and inferring TypeScript types from them
- Using the same schema on the client and the server
- Server actions for create, update, and delete
- useActionState, pending states, and toasts
- Accessible form errors and labels

In-class lab: Build the booking form (branch, doctor, date, slot, notes) and the admin doctor form with validation on both sides using server actions.

Session 2 TanStack Query and dashboard tables

- When to use TanStack Query in a Next.js app
- Queries, query keys, and caching
- Mutations, invalidation, and optimistic updates
- Search, filters, sorting, and pagination with URL search params
- Data tables with shadcn/ui Table
- Debouncing search input

In-class lab: Build the admin bookings table with search, status filter, sort, and pagination kept in the URL, plus cancel and confirm actions with optimistic updates.

Homework: Finish the My bookings page for patients with cancel and reschedule, and the admin services management page with create, edit, and delete.

Project milestone: Patients can book, cancel, and reschedule; admins can manage bookings, doctors, and services.

Week 7 Auth, Arabic and English, accessibility, and testing

Session 1 Authentication, roles, and i18n

- How sessions and tokens work, and where to store them safely
- Auth.js with credentials login against the Mawid API
- Protecting routes with middleware and checking roles on the server
- Admin, receptionist, and patient permissions
- next-intl: locale routes, message files, and a language switcher
- RTL layout with Tailwind logical utilities, Arabic dates and numbers

In-class lab: Add login and sign up, protect the dashboard by role, and translate Mawid to Arabic and English with a working RTL layout.

Session 2 Accessibility, testing, performance, and SEO

- Accessibility checks: keyboard navigation, focus, ARIA basics, and Lighthouse
- Vitest and Testing Library: testing components and hooks
- Testing forms and Zod schemas
- One Playwright end-to-end test for the booking flow
- Core Web Vitals: LCP, INP, CLS, next/image and next/font
- SEO: the Metadata API, Open Graph, sitemap, and robots

In-class lab: Write tests for the slot picker, the booking schema, and the booking form, add a Playwright test from login to confirmed booking, then fix the top Lighthouse issues.

Homework: Reach at least 10 passing tests, a Lighthouse accessibility score of 90 or more, full Arabic translation, and SEO metadata on every public page.

Project milestone: Mawid is feature complete: auth and roles, bilingual RTL, tests, and SEO, ready for deploy.

Building with AI · Weeks 8

Week 8 Building with AI the right way

Session 1 AI coding tools in a real project

- How Claude Code and Cursor work: context, prompts, and project rules files
- Writing a clear task: goal, files, constraints, and acceptance checks
- Asking for a plan before code, and working in small steps
- Reading a diff line by line before accepting it
- Common AI mistakes in React and Next.js: wrong client and server boundaries, outdated APIs, invented packages
- Keeping secrets and client data out of prompts

In-class lab: Use Claude Code or Cursor to add a doctor availability calendar to Mawid, review every diff, and reject or fix at least two wrong suggestions.

Session 2 Tests as a safety net and when not to trust AI

- Write the test first, then let the AI implement
- Using AI to write tests, and checking that the tests can fail
- Refactoring with AI while tests stay green
- Debugging with AI: give the error, the code, and what you tried
- When not to trust it: auth, payments, security, and licensing
- Explaining every line you ship in a code review

In-class lab: Write failing tests for a booking reminders feature, let the AI implement it until the tests pass, then do a pair code review and explain each change.

Homework: Ship one new Mawid feature with AI help through a pull request that includes the prompts used, the tests added, and a short note on what you rejected and why.

Project milestone: Mawid has an AI assisted feature delivered with tests and a reviewed pull request.

Capstone · Weeks 9

Week 9 Capstone build and real deploy

Session 1 Plan and build

- Picking a brief and cutting scope to fit one week
- Writing user stories and a page and route map
- Data model and API contract
- Repo setup: TypeScript, linting, Prettier, folder structure
- Daily plan and GitHub issues as a task board

In-class lab: Present your capstone plan in 5 minutes, get feedback, then build the core pages and the main flow in class with instructor check ins.

Session 2 Deploy to production

- Deploying to Vercel: projects, preview deployments, and environment variables
- Alternative: deploying to a VPS with Coolify
- Custom domain and HTTPS
- Production checks: error pages, logs, and basic monitoring
- Writing a strong README with screenshots and setup steps

In-class lab: Deploy your capstone and Mawid to production with environment variables and a custom domain or subdomain, then run your Playwright test against the live URL.

Homework: Finish the capstone: main flow working end to end, tests passing, live URL, README with screenshots, and a 3 minute demo video.

Project milestone: Capstone deployed and live with a public URL and a complete README.

Week 10 Career week and demo day

Session 1 CV, LinkedIn, GitHub, and interviews

- A one page frontend developer CV that passes ATS
- LinkedIn profile: headline, about section, featured projects
- GitHub portfolio: pinned repos, profile README, and live links
- Frontend interview topics: JavaScript, React, Next.js, CSS, and the browser
- Mock technical interview with live coding
- Freelancing first steps: Mostaql, Upwork, pricing a small project, and writing a proposal

In-class lab: Update your CV, LinkedIn, and GitHub profile in class with instructor review, then do a 30 minute mock technical interview in pairs.

Session 2 Demo day

- Structuring a 5 minute product demo
- Explaining your technical decisions and tradeoffs
- Answering questions from engineers
- Feedback from predev. Solutions engineers and guests
- Next steps plan and graduation

In-class lab: Present your capstone live to predev. Solutions engineers and guests, answer their questions, and get written feedback.

Homework: Apply to at least 10 frontend roles or freelance projects in the next two weeks and share progress in the alumni group.

Capstone

Each student builds and deploys their own web app from a brief chosen with the instructor. It must solve a real problem for Egyptian users, include login and a dashboard, and use the same stack as Mawid.

Requirements

- Next.js 15 App Router with TypeScript, Tailwind CSS 4, and shadcn/ui
- Login with at least two user roles and protected pages
- At least one create, edit, and delete flow with React Hook Form and Zod validation
- A dashboard page with a table that has search, filters, and pagination
- Arabic and English with correct RTL layout
- Loading, empty, and error states on every data page
- At least 8 unit or component tests and one Playwright end-to-end test
- SEO metadata on public pages and a Lighthouse accessibility score of 90 or more
- Deployed to Vercel or a VPS with Coolify, with environment variables and a live URL
- Clean Git history and a README with screenshots, setup steps, and the live link

Example ideas

- Gym membership and class booking with a trainer dashboard
- Private tutoring center: groups, sessions, and attendance for teachers and parents
- Restaurant table reservations with an admin view for the branch manager
- Car service center: book a service slot and track the car status
- Real estate listings for Egypt compounds with an agent dashboard
- Freelancer invoices and client tracker with payment status

Capstone rubric

Works end to end on the live URL	25%
Code quality: structure, TypeScript, components, server and client split	20%
UI and UX: responsive, Arabic RTL, accessibility	15%
Forms, data, and auth handled correctly	15%
Tests: unit, component, and one end-to-end test	10%
Deploy, README, and Git history	10%
Demo day presentation and answers	5%

How you are assessed

Weekly homework Submitted as GitHub pull requests, reviewed against a short checklist, late work gets half marks	20%
Running project milestones Mawid milestones in weeks 4-8, checked in class against the feature list	25%
Quizzes and live code reviews Short quizzes in weeks 3, 5, and 7, plus explaining your own code in review	10%
AI week assignment Pull request with prompts, tests, and notes on rejected suggestions	5%
Capstone project Graded with the capstone rubric	30%
Career week deliverables Updated CV, LinkedIn, GitHub profile, and mock interview	10%

Skills checklist

HTML and CSS

- [ ] Writes semantic HTML with correct headings, labels, and alt text
- [ ] Builds responsive layouts with Flexbox and Grid, mobile first
- [ ] Uses logical properties so layouts work in RTL and LTR

JavaScript and TypeScript

- [ ] Uses array methods, destructuring, spread, and modules
- [ ] Writes async code with async and await and handles errors
- [ ] Types props, API responses, and functions with TypeScript

React

- [ ] Splits a UI into small reusable components with typed props
- [ ] Manages state, effects, and context, and writes custom hooks
- [ ] Avoids unneeded effects and re-renders

Next.js

- [ ] Structures routes, layouts, and loading and error files in the App Router
- [ ] Chooses between server and client components
- [ ] Fetches data on the server and uses server actions for mutations
- [ ] Understands caching and revalidation basics

Forms, data, and auth

- [ ] Builds forms with React Hook Form and Zod on client and server
- [ ] Uses TanStack Query for queries, mutations, and invalidation
- [ ] Protects routes and checks roles on the server

Quality and delivery

- [ ] Writes Vitest and Testing Library tests and one Playwright test
- [ ] Checks accessibility, Core Web Vitals, and SEO metadata
- [ ] Uses Git branches and pull requests, and deploys to Vercel or Coolify
- [ ] Uses AI tools with review and tests, and can explain every line

After graduation

Months 1-3 after graduation

- Apply for junior frontend and internship roles every week
- Polish Mawid and your capstone, and add one more small project
- Practice JavaScript and React interview questions and small coding problems
- Take one small freelance job to get a first client review

Months 4-6

- Learn a backend for your frontend: Node.js 24 LTS with a REST API and PostgreSQL 17
- Go deeper in Next.js caching, server actions, and performance
- Contribute small fixes to an open source React project
- Learn state management for larger apps, such as Zustand

Months 7-12: toward mid-level

- Own features end to end in a team: planning, code, tests, and deploy
- Learn CI basics, code review habits, and monitoring in production
- Explore React Native if you want to add mobile
- Start mentoring juniors and writing short technical posts on LinkedIn

Where this track leads

Junior Frontend Developer (React) | Junior Next.js Developer | React Developer at software houses and startups | Frontend Intern or Trainee | Freelance Web Developer on Mostaql and Upwork | Remote Junior Frontend Developer for Gulf and European companies | Junior Full Stack Developer after adding a backend

Resources

- MDN Web Docs: HTML: https://developer.mozilla.org/en-US/docs/Web/HTML
- MDN Web Docs: CSS: https://developer.mozilla.org/en-US/docs/Web/CSS
- MDN Web Docs: JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- The Modern JavaScript Tutorial: https://javascript.info/
- TypeScript Handbook: https://www.typescriptlang.org/docs/
- React docs: https://react.dev/learn
- Next.js docs: https://nextjs.org/docs

- Next.js Learn course: https://nextjs.org/learn
- Tailwind CSS docs: https://tailwindcss.com/docs
- shadcn/ui docs: https://ui.shadcn.com/docs
- React Hook Form: https://react-hook-form.com/
- Zod: https://zod.dev/
- TanStack Query: https://tanstack.com/query/latest
- Auth.js: https://authjs.dev/
- next-intl: https://next-intl.dev/
- Vitest: https://vitest.dev/
- React Testing Library: https://testing-library.com/docs/react-testing-library/intro/
- Playwright: https://playwright.dev/
- web.dev: Core Web Vitals: https://web.dev/articles/vitals
- W3C: Introduction to Web Accessibility: https://www.w3.org/WAI/fundamentals/accessibility-intro/
- Git documentation: https://git-scm.com/doc
- GitHub Docs: https://docs.github.com/en
- Vercel docs: https://vercel.com/docs
- Coolify docs: https://coolify.io/docs/
- Claude Code docs: https://docs.anthropic.com/en/docs/claude-code/overview