CodeSquadz

React.js

6-Week Job-Ready Course

A practical 1.5-month roadmap with 3 hands-on projects

Table of Contents

1. How to use this roadmap
2. Pre-requisites checklist
3. Week 1 — JavaScript refresh + React fundamentals
4. Week 2 — Hooks, events, forms & lifecycle
5. Project 1 — Personal Portfolio + Todo Manager
6. Week 3 — Routing, Context API & advanced hooks
7. Week 4 — API integration & state management
8. Project 2 — Movie Discovery App (TMDB API)
9. Week 5 — Performance, Framer Motion, styling, testing & TypeScript
10. Project 3 — Animated Analytics Dashboard
11. Week 6 — Auth, deployment, capstone & interview prep

How to use this roadmap?

This is a 6-week, 6-day-per-week plan with 3 dedicated projects placed in between the weeks (not just at the end). Sundays are for revision, polish, and applying to jobs. Each project takes 2-3 days and goes straight to your GitHub. The goal isn't memorizing APIs — it's building muscle memory by typing code daily.

Pre-requisites checklist

- HTML5 & CSS3 (Flexbox, Grid, responsive design)
- JavaScript fundamentals (variables, functions, objects, arrays, DOM)
- Node.js + npm installed (Node 18 or higher recommended)
- Git & GitHub basics (clone, commit, push, branch, pull request)
- VS Code with extensions: ES7+ React snippets, Prettier, ESLint
- Browser DevTools comfort (Console, Network, Elements tabs)

Week 1 — JavaScript Refresh + React Fundamentals

Goals

- Lock in modern JS (ES6+) — destructuring, spread, modules, async
- Understand what React is and why it exists
- Build your first components with JSX and props
- Set up a clean Vite + React development environment

Daily breakdown

Day	Topics & Practice
Day 1	• ES6+ recap: let/const, arrow functions, template literals • Destructuring (objects + arrays), spread & rest operators • Practice: rewrite 10 ES5 functions in ES6 style
Day 2	• Array methods: map, filter, reduce, find, some, every • Optional chaining (?.) and nullish coalescing (??) • Practice: 15 array transformation problems
Day 3	• Promises, async/await, fetch API • ES Modules (import/export, default vs named) • Mini-task: fetch & display data from JSONPlaceholder API
Day 4	• What is React? Virtual DOM, declarative vs imperative • Setup: npm create vite@latest (Vite is preferred over CRA in 2026) • Folder structure, entry point (main.jsx), root mount
Day 5	• JSX deep dive: expressions, attributes, fragments • Functional components, exporting, importing • Build: Profile Card component (avatar, name, bio)
Day 6	• Props, prop drilling, prop types (basic) • Children prop, composition pattern • Mini-task: Recipe card grid (props-driven, 6+ cards)

Week 2 — Hooks, Events, Forms & Lifecycle

Goals

- Master useState and useEffect — interview must-knows
- Handle events, controlled forms, and conditional UI
- Render dynamic lists with proper keys
- Understand component lifecycle in functional components

Daily breakdown

Day	Topics & Practice
Day 1	• useState hook: primitives, objects, arrays • Functional updates: setCount(c => c + 1) • Build: Counter app with increment, decrement, reset
Day 2	• Event handling (onClick, onChange, onSubmit) • Synthetic events, e.preventDefault(), e.target.value • Conditional rendering (&&, ternary, early return)
Day 3	• Lists & keys, why index-as-key is dangerous • Lifting state up, sibling communication • Practice: render todos from an array, add/delete/toggle
Day 4	• Controlled vs uncontrolled inputs • Form handling: text, checkbox, radio, select, textarea • Build: contact form with basic validation
Day 5	• useEffect: dependency array, cleanup function • Common patterns: data fetch, event listeners, timers • Common bugs: infinite loops, stale closures
Day 6	• Combine all Week 1+2 concepts into one mini-app • Practice debugging useEffect issues • Get ready for Project 1 (next 2-3 days)
Interview alert: 'Explain useEffect dependencies' is asked in almost every React interview. Practice explaining it out loud — empty array, missing deps, cleanup — until it's reflex.	

CodeSquadz

Daily breakdown

Day	Topics & Practice
Day 1	• React Router v6: BrowserRouter, Routes, Route, Link, NavLink • useNavigate, useLocation, useParams • Build: 3-page site (Home, About, Contact)
Day 2	• Nested routes, Outlet, dynamic params (/users/:id) • 404 / Not Found page, redirect patterns • Protected routes (basic auth gate)
Day 3	• useRef: DOM refs, persisting values without re-render • Forwarding refs (forwardRef) — common interview topic • Practice: focus input on mount, scroll-to-top button
Day 4	• useMemo & useCallback: when to use, when NOT to use • Reference equality, why React re-renders • Common myth: 'wrap everything in memo' is wrong
Day 5	• Context API: createContext, Provider, useContext • useReducer: when state logic gets complex • Combine Context + useReducer for a mini Redux
Day 6	• Custom hooks: useFetch, useLocalStorage, useDebounce • Refactor your Week 2 todo app to use custom hooks • Add routing to make portfolio multi-page

CodeSquadz

Week 4 — API Integration & State Management

Goals

- Fetch data the right way — loading, error, empty states
- Use TanStack Query (React Query) — industry standard for server state
- Pick up Redux Toolkit basics (still asked in 60%+ interviews)
- Try Zustand for lightweight global state

Daily breakdown

Day	Topics & Practice
Day 1	• Fetch API vs Axios — interceptors, defaults • useEffect data fetching pattern + AbortController • Loading / error / success state machine
Day 2	• TanStack Query: useQuery, useMutation • Caching, refetching, stale time, query keys • Why React Query beats useEffect for server data
Day 3	• Pagination, infinite scroll with useInfiniteQuery • Optimistic updates pattern • Practice: GitHub user search with debounce + caching
Day 4	• Redux Toolkit: store, slices, createSlice, configureStore • useSelector, useDispatch, immer-powered reducers • createAsyncThunk for async actions
Day 5	• Zustand basics — much less boilerplate • When to pick Context vs Redux vs Zustand vs Query • Mental model: server state vs client state
Day 6	• Connect your portfolio to a real API • Practice combining Redux Toolkit + React Query • Get ready for Project 2 (next 3 days)

Reality check: Modern React jobs rarely write reducers by hand. Learn Redux Toolkit, not classic Redux. But still understand the core concepts — interviews love the 'what is reducer purity' question.

PROJECT 2 Movie Discovery App (TMDB API)
Build after Week 4 • Time: 3 days

Goal

Build a real-world app that fetches data from a public API, handles routing, manages global state, and feels production-grade. This project shows you can integrate APIs and structure a multi-page React app.

Core Features to Build

- Home page: Trending, top-rated, and now-playing movie carousels
- Search page: Debounced search with TMDB API + paginated results
- Movie detail page: Dynamic route /movie/:id with full info, cast, trailer
- Watchlist: Add/remove movies to watchlist (Redux Toolkit or Zustand)
- Genre filter: Filter movies by genre using URL query params
- Loading skeletons: Show shimmer placeholders while data fetches
- Error handling: Friendly error messages on API failures
- Persisted watchlist: Save to localStorage so it survives refresh

Tech Stack

- Vite + React, React Router v6, TanStack Query (for API caching), Redux Toolkit or Zustand (for watchlist), Axios, plain CSS or CSS Modules. Get a free API key from themoviedb.org.

Deliverable

- Deployed link + GitHub repo. Add a demo GIF in the README showing search, browse, and watchlist features. This project alone can land interview calls — make it shine.

Week 5 — Performance, Framer Motion, Styling, Testing & TypeScript

Goals

- Optimize re-renders & bundle size (real perf, not premature)
- Add professional animations using Framer Motion
- Style apps with Tailwind CSS + a component library
- Write tests with React Testing Library
- Get comfortable with TypeScript in React (most jobs require it)

Tech Stack

- Vite + React, React Router v6, TanStack Query (for API caching), Redux Toolkit or Zustand (for watchlist), Axios, plain CSS or CSS Modules. Get a free API key from themoviedb.org.

Deliverable

- Deployed link + GitHub repo. Add a demo GIF in the README showing search, browse, and watchlist features. This project alone can land interview calls — make it shine.

Daily breakdown

Day	Topics & Practice
Day 1	• React.memo, when memoization actually helps • Lazy loading with React.lazy + Suspense • Code splitting routes, dynamic imports • Error boundaries (class component — yes, still needed)
Day 2	• **Framer Motion intro:** motion components (motion.div, motion.button) • Animate prop, initial & animate states, transition options • Hover, tap, drag gestures (whileHover, whileTap, drag) • Practice: animated buttons, card hover effects, page entrance
Day 3	• **Framer Motion advanced:** AnimatePresence for enter/exit animations • Variants for orchestrated animations & staggered children • Layout animations (layoutId for shared element transitions) • Scroll-triggered animations with useScroll & useTransform
Day 4	• Tailwind CSS: setup with Vite, utility-first mindset • shadcn/ui — copy-paste component library trend in 2025-26 • Combine Tailwind classes with Framer Motion smoothly • Redesign your portfolio with Tailwind + animations
Day 5	• React Hook Form: register, handleSubmit, validation • Zod schemas for type-safe validation • TypeScript with React: typing props, useState, useRef • Generic components, event types, common pitfalls
Day 6	• Vitest + React Testing Library setup • Unit tests for components, user-event interactions • Convert one of your JSX projects to TSX • Get ready for Project 3 (next 2-3 days)

CodeSquadz

Why Framer Motion matters: Smooth animations are what separate a "junior portfolio" from a "hireable portfolio". Recruiters scroll through dozens of projects daily — well-animated UIs immediately stand out and signal attention to detail. Framer Motion is also the most asked animation library in React frontend interviews.

PROJECT 3 Animated Analytics Dashboard
Build after Week 5 • Time: 2-3 days

Goal

- Build a polished, animated dashboard in TypeScript that demonstrates your skill with charts, animations, modern styling, and forms. This is the project that goes at the top of your portfolio because it screams "I can build production UI."

Core Features to Build

- Sidebar navigation: Animated active state with Framer Motion layoutId
- KPI cards: Animate numbers counting up on mount (use Framer Motion)
- Charts: Line, bar, and pie charts using Recharts (revenue, users, etc.)
- Data table: Sortable, filterable, paginated table of transactions
- Animated modals: Slide-in modal for adding/editing entries (AnimatePresence)
- Theme toggle: Dark/light mode with smooth color transition
- Form with validation: Add new entries using React Hook Form + Zod
- Page transitions: Smooth route transitions between dashboard sections
- Tested: Write at least 5 unit tests for key components

Tech Stack

- Vite + React + TypeScript, Tailwind CSS, shadcn/ui components, Framer Motion for animations, Recharts for charts, React Hook Form + Zod for forms, Vitest + RTL for tests. Use mock JSON data or json-server.

Deliverable

- Deployed link + GitHub repo with comprehensive README. Record a 30-second screen recording showing the animations and embed it in the README. This project should feel like a real SaaS dashboard.

PROJECT 1 Personal Portfolio + Todo Manager
Build after Week 2 • Time: 2-3 days

Goal

Combine everything from Weeks 1-2 into a single deployed app. This will be your first real React project on GitHub and will showcase your understanding of components, props, state, hooks, events, and forms.

Core Features to Build

- Portfolio section: Hero, About, Skills grid, Projects list (props-driven)
- Todo manager: Add, edit, delete, mark complete, filter (all/active/done)
- localStorage persistence: Todos survive page refresh
- Dark/light mode toggle: Use useEffect to update body class
- Form validation: Don't allow empty todos, max 100 chars
- Responsive design: Works on mobile, tablet, desktop
- Bonus: Add a simple counter widget to flex useState skills

Tech Stack

- Vite + React (JSX), CSS Modules or plain CSS, localStorage API. No external libraries yet — keep it pure React.

Deliverable

- Live deployed link on Vercel + clean GitHub repo with proper README (project description, screenshots, tech stack, live link, how to run locally). This is the first project recruiters will see — polish it.

Week 3 — Routing, Context API & Advanced Hooks

Goals

- Build SPAs with React Router v6 (the current standard)
- Share state across components without prop drilling
- Use useRef, useMemo, useCallback correctly (not blindly)
- Write your own custom hooks

Goals

- Implement JWT auth on the frontend (login, protected routes)
- Deploy properly with environment variables
- Polish resume, GitHub, LinkedIn — and start applying

Daily breakdown

Day	Topics & Practice
Day 1	• JWT auth flow: login, store token, attach to requests • Axios interceptors for auth headers + 401 handling • Protected routes pattern with React Router
Day 2	• Environment variables (VITE_API_URL etc.) • Deploy frontend to Vercel, backend to Render/Railway • CORS, production gotchas, build errors debugging
Day 3	• Next.js intro (App Router) — 1 day taste, not full dive • When to choose Next.js over Vite + React • SEO with React Helmet for SPAs
Day 4-5	• Capstone Project — pick ONE and ship hard: • Option A: Job tracker app (auth, CRUD, filters, Kanban view, animations) • Option B: Notion-like note app (rich text, folders, search) • Option C: Real-time chat (Socket.io + auth + groups) • Option D: AI chatbot UI hooked to Anthropic API (your stack!) • Use everything: TS + Tailwind + Framer Motion + React Query + auth
Day 6	• Resume polish: 1 page, project-first, metrics where possible • GitHub README templates, pin best 6 repos • LinkedIn headline + projects section + start applying

A Steps towards the bright future
If you required any further information, please fill free to contact :

CodeSquadz

For more Information, Contact

78-79-33-22-11

+91-9289135096

enquiry@codesquadz.com

www.codesquadz.com