# Preliminary Tech Stack for React Learning Project

This document outlines the preliminary technology stack and design principles for the React learning project. The goal is to build a visually polished, testable app that consumes an API, without relying on high-level abstractions. This stack is subject to change as the project evolves.

## Core Build Tools

- Vite + React + TypeScript

- ESLint + Prettier for code quality

- Husky + lint-staged for pre-commit checks

## Styling

- Tailwind CSS for utility-first design

- Optional: daisyUI or shadcn/ui for component inspiration

- Animations via CSS transitions or reactbits.dev ideas

- Optional: Framer Motion for targeted advanced interactions

## Forms

- Hand-built controlled components

- No react-hook-form or Formik

- Optional validation with Zod

## API Handling

- Native fetch with AbortController

- Custom lightweight API wrapper

- Custom hooks (e.g., useApi) for fetching and error/loading states

- No TanStack Query or SWR (keep low-level for learning)

## State Management

- Local component state

- Optional React Context for global values (e.g., theme)

- No Redux/MobX unless truly needed

## Testing

- Vitest + React Testing Library

- MSW (Mock Service Worker) for mocking API calls

- Playwright (optional) for end-to-end testing

## Design Guidelines

- Semantic design tokens (colors, spacing, typography)

- Consistent layout rhythm with 12-column grid

- Micro-interactions: hover, focus, subtle shadows

- Accessible: focus states, contrast, semantic HTML

## Inspiration Sources

- UI flows: Mobbin, Pageflows, Pttrns

- Visual polish: Awwwards, Godly, Lapa Ninja

- Components: shadcn/ui, Tailwind UI

- Colors & Type: coolors.co, Google Fonts, Type-Scale.com

- Empty states: emptystat.es

## Project Phases

- Scaffold & theming

- Routing & layout shell

- API baseline (list + details with loading/error states)

- Forms (controlled, inline validation)

- Lists & filtering with API integration

- Polish with motion and skeletons

- Testing (unit, integration, e2e)

- Performance tweaks (code-splitting, prefetching)