

# Building Simple Web Apps with AI Tools

Lucas Soares, Instructor

September 30, 2025

# Course Agenda

## **1. Methodology Notes**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**

# Course Agenda

1. **Methodology Notes**
2. **What this Course is and What is Not**
3. **Motivation**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**
- 7. Hands-on: Building A Simple Webpage**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**
- 7. Hands-on: Building A Simple Webpage**
- 8. Notes on Best Practices**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**
- 7. Hands-on: Building A Simple Webpage**
- 8. Notes on Best Practices**
- 9. Hands-on: Building a Quiz App**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**
- 7. Hands-on: Building A Simple Webpage**
- 8. Notes on Best Practices**
- 9. Hands-on: Building a Quiz App**
- 10. Extending capabilities with APIs**

# Course Agenda

- 1. Methodology Notes**
- 2. What this Course is and What is Not**
- 3. Motivation**
- 4. AI Tools**
- 5. Workflow for Building Apps**
- 6. Hands-on: Cursor/Claude Walkthrough**
- 7. Hands-on: Building A Simple Webpage**
- 8. Notes on Best Practices**
- 9. Hands-on: Building a Quiz App**
- 10. Extending capabilities with APIs**
- 11. Hands-on: Interactive Live Coding**

# Methodology Notes

1. **Presentation:** Theoretical concepts and explanations

# Methodology Notes

1. **Presentation:** Theoretical concepts and explanations
2. **Demonstration:** Live walkthrough of concepts

# Methodology Notes

1. **Presentation:** Theoretical concepts and explanations
2. **Demonstration:** Live walkthrough of concepts
3. **Recap Summary:** Key takeaways and discussion

# Methodology Notes

1. **Presentation:** Theoretical concepts and explanations
2. **Demonstration:** Live walkthrough of concepts
3. **Recap Summary:** Key takeaways and discussion
4. **Interactive Q&A:** Addressing specific questions

# Methodology Notes

1. **Presentation:** Theoretical concepts and explanations
2. **Demonstration:** Live walkthrough of concepts
3. **Recap Summary:** Key takeaways and discussion
4. **Interactive Q&A:** Addressing specific questions
5. **Break:** Time to process and reflect

# What This Course Is and Is Not

## NOT About:

- Building commercial applications
- Deploying apps for profit
- Complex software architecture

# What This Course Is and Is Not

## NOT About:

- Building commercial applications
- Deploying apps for profit
- Complex software architecture

## IS About:

- Learning how to be more productive with AI tools
- Leveraging AI tools effectively to build apps for yourself
- Using pure HTML/JavaScript for personal apps without hassle
- Building tools for your own workflow needs
- Understanding AI-assisted development

# Course Overview

This course will empower you to:

- Build functional web applications using AI assistance

# Course Overview

This course will empower you to:

- Build functional web applications using AI assistance
- Create and refine technical specifications efficiently

# Course Overview

This course will empower you to:

- Build functional web applications using AI assistance
- Create and refine technical specifications efficiently
- Develop effective problem-solving approaches

# Course Overview

This course will empower you to:

- Build functional web applications using AI assistance
- Create and refine technical specifications efficiently
- Develop effective problem-solving approaches
- Deploy personal applications that address local needs

# Course Overview

This course will empower you to:

- Build functional web applications using AI assistance
- Create and refine technical specifications efficiently
- Develop effective problem-solving approaches
- Deploy personal applications that address local needs

*The goal isn't to turn you into a professional developer, but to give you the tools and confidence to create solutions for problems that matter to you.*

# Target Audience

- **Technically curious** non-developers

# Target Audience

- **Technically curious** non-developers
- Low-code/no-code tool **power users**

# Target Audience

- **Technically curious** non-developers
- Low-code/no-code tool **power users**
- Professionals looking to **solve specific problems**

# Target Audience

- **Technically curious** non-developers
- Low-code/no-code tool **power users**
- Professionals looking to **solve specific problems**
- **Community builders** with specific software needs

# Target Audience

- **Technically curious** non-developers
- Low-code/no-code tool **power users**
- Professionals looking to **solve specific problems**
- **Community builders** with specific software needs

If you've ever thought, *"I wish there was an app for that."* this course is for you.

# Tools and Resources Needed

To participate fully in this course, you'll need:

- Computer with browser and internet connection
- Free accounts for Cursor IDE and Claude
- Optional: GitHub account

# Tools and Resources Needed

To participate fully in this course, you'll need:

- Computer with browser and internet connection
- Free accounts for Cursor IDE and Claude
- Optional: GitHub account

All the tools we'll use have **free tiers** sufficient for our coursework.

# Motivation

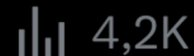
Home-cooked Software and Barefoot Programmers: Maggie Ap...





**Simon Willison**  @simonw · 13h

Content like this might be better served by bundling all of the assets into a single HTML file (using base64-encoded images, inline CSS etc) - effectively using HTML as a single file format (like a PDF) for improved longevity and shareability



[Simon Willison Twitter](#)

# Why Learn to Build Apps with AI?

- **AI tools lower the barrier to software creation**

# Why Learn to Build Apps with AI?

- AI tools lower the barrier to software creation
- Build **personal, local-first applications** that solve real problems

# Why Learn to Build Apps with AI?

- **AI tools lower the barrier to software creation**
- Build **personal, local-first applications** that solve real problems
- No professional development experience required

# Why Learn to Build Apps with AI?

- **AI tools lower the barrier to software creation**
- Build **personal, local-first applications** that solve real problems
- No professional development experience required
- Gain **practical skills** for leveraging AI in software development

# Why Learn to Build Apps with AI?

- **AI tools lower the barrier to software creation**
- Build **personal, local-first applications** that solve real problems
- No professional development experience required
- Gain **practical skills** for leveraging AI in software development
- Move beyond reliance on big tech by creating **your own tools**

# Who are Barefoot Developers?

# Who are Barefoot Developers?

- Technically curious but not full-time programmers

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use
- Solve **specific problems** that commercial software ignores

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use
- Solve **specific problems** that commercial software ignores

## Why this mindset matters

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use
- Solve **specific problems** that commercial software ignores

## Why this mindset matters

- Empowers individuals to take control of their digital tools

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use
- Solve **specific problems** that commercial software ignores

## Why this mindset matters

- Empowers individuals to take control of their digital tools
- Encourages practical experimentation with AI

# Who are Barefoot Developers?

- Technically curious but not full-time programmers
- Build software for personal & community use
- Solve **specific problems** that commercial software ignores

## Why this mindset matters

- Empowers individuals to take control of their digital tools
- Encourages practical experimentation with AI
- Bridges the gap between no-code and full software development

# AI Tools

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models (Claude, ChatGPT, Gemini...)**

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models** (Claude, ChatGPT, Gemini...)
2. **AI-enhanced IDEs** (Cursor)

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models** (Claude, ChatGPT, Gemini...)
2. **AI-enhanced IDEs** (Cursor)
3. **Frontend technologies** (HTML, CSS, JavaScript)

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models** (Claude, ChatGPT, Gemini...)
2. **AI-enhanced IDEs** (Cursor)
3. **Frontend technologies** (HTML, CSS, JavaScript)
4. **Storage options** (everything stored locally)

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models (Claude, ChatGPT, Gemini...)**
2. **AI-enhanced IDEs (Cursor)**
3. **Frontend technologies (HTML, CSS, JavaScript)**
4. **Storage options (everything stored locally)**
5. **Deployment methods**

# Core Components Overview

Our development stack consists of five key elements:

1. **Large Language Models** (Claude, ChatGPT, Gemini...)
2. **AI-enhanced IDEs** (Cursor)
3. **Frontend technologies** (HTML, CSS, JavaScript)
4. **Storage options** (everything stored locally)
5. **Deployment methods**

This combination of tools enables us to build complete applications with minimal technical background.

# Hands-on: Claude Walkthrough

# Q&A & Break

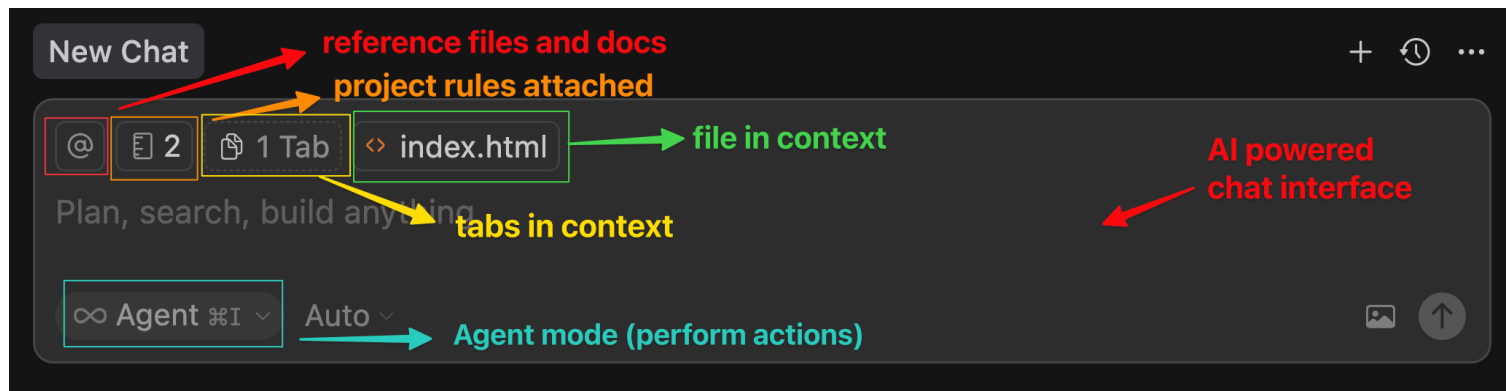
# Claude Recap

- **Multi-Modal Chat:** General-purpose multi-modal communication
- **Context:** 200k+ tokens of context length (~500 pages)
- **Code Generation:** Writes code based on the context
- **Code Review:** Helps refine code
- **Claude Projects:** Work on multiple files at once for complex projects
- **Claude Artifacts:** Run and preview code in the browser
- **Prompting tips:** use role, give context, be clear, specific and direct
- **Meta Prompt:** ask claude to improve your own prompt

# Hands-on: Cursor Walkthrough

# Q&A & Break

# Cursor IDE Recap



## MCP

### Installed Servers

C

context7

`npx -y @upstash/context7-mcp`

resolve-library-id

get-library-docs

Show less ▾

+

New MCP Server

Add a Custom MCP Server

# Cursor IDE Recap

- AI-enhanced code editor

# Cursor IDE Recap

- AI-enhanced code editor
- Spec document capabilities

# Cursor IDE Recap

- AI-enhanced code editor
- Spec document capabilities
- Cursor rules for consistent development
  - You can generate automatically with `/Generate Cursor Rules`

# Cursor IDE Recap

- AI-enhanced code editor
- Spec document capabilities
- Cursor rules for consistent development
  - You can generate automatically with `/Generate Cursor Rules`
- File referencing with `@` symbol for context

**Prompt for a welcome page to this course**

@ 2 1 Tab presentation.html

Prompt for Cursor

Build me a welcome page for this course I'm working on (presentation slides are in @presentation.html )

Make the webpage in pure .html and javascript within a single file, make it sleek, modern minimalistic design and some cool functional interactive features.

∞ Agent %I Auto

I'll create a sleek, modern welcome page for your course based on the presentation content. Let me analyze the presentation first to understand the course structure and then build an interactive welcome page.

To-dos 4

Analyze presentation.html to understand course content and structure

Design modern, minimalistic welcome page with interactive features

Add cool interactive features like animations, hover effects, and dynamic content

Build complete welcome page as single HTML file with embedded CSS and JavaScript

@ 2 1 Tab

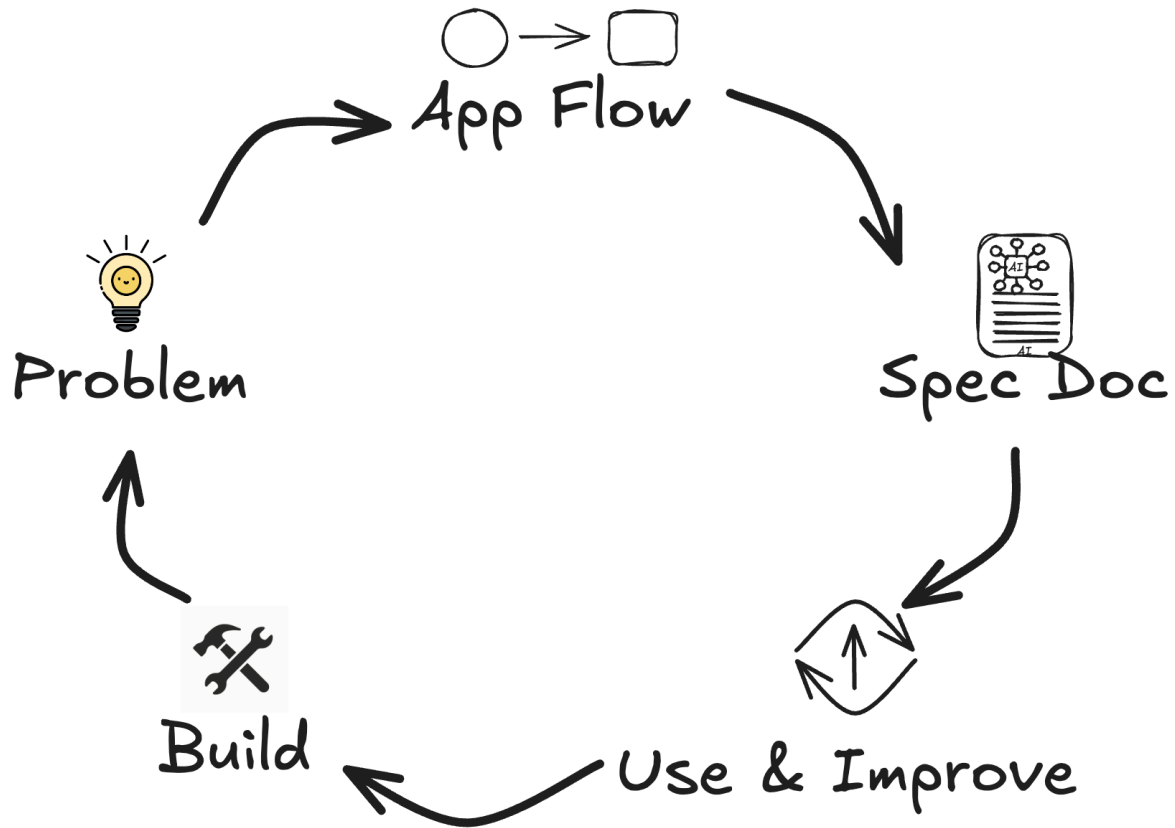
29.7%

Plan, search, build anything

∞ Agent %I Auto

# Welcome Page Slide

# Workflow for Building Apps



Our approach is **iterative and AI-collaborative**,  
focusing on continuous refinement based on user  
feedback.

# Whiteboard Session

# Hands-on: Building a Simple Webpage

# Q&A & Break

# Notes on Best Practices

# Problem

# Problem

## Brainstorm with Claude




- Explore problem space
- Generate initial ideas
- Refine concepts through dialogue

# Problem

## Brainstorm with Claude

- Explore problem space
- Generate initial ideas
- Refine concepts through dialogue

## Research with AI Tools

- Perplexity 
- ChatGPT 
- Gemini 

# App Flow

## Planning

# App Flow

## Planning

- Use voice mode to communicate faster

# App Flow

## Planning

- Use voice mode to communicate faster
- The core idea

# App Flow

## Planning

- Use voice mode to communicate faster
- The core idea
- Essential features

# App Flow

## Planning

- Use voice mode to communicate faster
- The core idea
- Essential features
- The app flow (pages, navigation, user actions)

# App Flow

## Planning

- Use voice mode to communicate faster
- The core idea
- Essential features
- The app flow (pages, navigation, user actions)


## Diagramming Tools

# App Flow

## Planning


- Use voice mode to communicate faster
- The core idea
- Essential features
- The app flow (pages, navigation, user actions)

## Diagramming Tools


- Sketch the raw app flow 
  - Excalidraw
  - Figma (or similar)
  - Claude to generate mermaid diagrams

# Writing Detailed Spec Docs


# Writing Detailed Spec Docs

- Write detailed Product Requirements Document files 
  - PRD.md files
  - Project description + core functionalities
  - File structure
  - Tech stack overview (HTML/JS...)


# Writing Detailed Spec Docs

- Write detailed Product Requirements Document files 
  - PRD.md files
  - Project description + core functionalities
  - File structure
  - Tech stack overview (HTML/JS...)
- **Task Breakdown:** Break features into smallest possible tasks
  - Clear, actionable items for AI assistance
  - Optionally: use a `task_breakdown.md` for step-by-step implementation

# Writing Detailed Spec Docs

- Write detailed Product Requirements Document files 
  - `PRD.md` files
  - Project description + core functionalities
  - File structure
  - Tech stack overview (HTML/JS...)
- **Task Breakdown:** Break features into smallest possible tasks
  - Clear, actionable items for AI assistance
  - Optionally: use a `task_breakdown.md` for step-by-step implementation
- Control Cursor's agents behavior
  - Cursor rules `.mdc` files

# Writing Detailed Spec Docs

- Write detailed Product Requirements Document files 
  - PRD.md files
  - Project description + core functionalities
  - File structure
  - Tech stack overview (HTML/JS...)
- **Task Breakdown:** Break features into smallest possible tasks
  - Clear, actionable items for AI assistance
  - Optionally: use a task\_breakdown.md for step-by-step implementation
- Control Cursor's agents behavior
  - Cursor rules .mdc files
  - AGENTS.md files in Cursor

# Writing Detailed Spec Docs

- **Project Status Tracking:** `Project_Status.md` for progress
  - Update at end of each session
  - Resume seamlessly next time

# Writing Detailed Spec Docs

- **Project Status Tracking:** `Project_Status.md` for progress
  - Update at end of each session
  - Resume seamlessly next time
  - Leverage context with @ symbol for referencing files and/or indexed docs

# Writing Detailed Spec Docs

- **Project Status Tracking:** `Project_Status.md` for progress
  - Update at end of each session
  - Resume seamlessly next time
  - Leverage context with @ symbol for referencing files and/or indexed docs
  - Leverage MCP for adding functionality to Cursor's agents

# Writing Detailed Spec Docs

- **Project Status Tracking:** `Project_Status.md` for progress
  - Update at end of each session
  - Resume seamlessly next time
  - Leverage context with `@` symbol for referencing files and/or indexed docs
  - Leverage MCP for adding functionality to Cursor's agents
- **Documentation Sync:** Use `@Docs` for official documentation
  - Ensures accurate, up-to-date references

# Build

- Prototype with Claude Artifacts/Build through dialog with Cursor

# Build

- Prototype with Claude Artifacts/Build through dialog with Cursor
- **Structured Implementation:**
  - Follow task breakdown step by step (optionally: `task_breakdown.md`)
  - Tackle features one by one from the breakdown
  - Reference context files for each task

# Build

- Prototype with Claude Artifacts/Build through dialog with Cursor
- **Structured Implementation:**
  - Follow task breakdown step by step (optionally: `task_breakdown.md`)
  - Tackle features one by one from the breakdown
  - Reference context files for each task
- **UI Refinement:** Use design inspiration
  - Upload screenshots from websites/apps you like (tools: Dribbble/Behance)
  - Prompt: "Replicate this design, keeping current copy and colors"

# Build

- Prototype with Claude Artifacts/Build through dialog with Cursor
- **Structured Implementation:**
  - Follow task breakdown step by step (optionally: `task_breakdown.md`)
  - Tackle features one by one from the breakdown
  - Reference context files for each task
- **UI Refinement:** Use design inspiration
  - Upload screenshots from websites/apps you like (tools: Dribbble/Behance)
  - Prompt: "Replicate this design, keeping current copy and colors"
- **Progress Tracking:** Update `Project_Status.md`
  - End each session with status update
  - Resume next time: "Read Project Status and continue"

# Key Workflow Principles

## Plan Everything Upfront

- Clear PRD, task breakdown, and context files
- Structured approach reduces AI hallucinations
- Better output accuracy with proper context

# Key Workflow Principles

## Plan Everything Upfront

- Clear PRD, task breakdown, and context files
- Structured approach reduces AI hallucinations
- Better output accuracy with proper context

## Use AI Tools' Strengths

- Combine AI assistance with manual coding
- Leverage context files and documentation sync
- Iterate effectively with design inspiration

# Maintain Context Continuity

- Project Status tracking between sessions
- Clear file responsibilities and references
- Seamless development process

# Maintain Context Continuity

- Project Status tracking between sessions
- Clear file responsibilities and references
- Seamless development process

# Focus on Quality

- Review AI-generated code thoroughly
- Test across different scenarios
- Refine UI with external inspiration

The reality of building web apps in 2025 is that it's a bit like assembling IKEA furniture. There's no "full-stack" product with batteries included, you have to piece together and configure many individual services:

- frontend / backend (e.g. React, Next.js, APIs)
- hosting (cdn, https, domains, autoscaling)
- database
- authentication (custom, social logins)
- blob storage (file uploads, urls, cdn-backed)
- email
- payments
- background jobs
- analytics
- monitoring
- dev tools (CI/CD, staging)
- secrets
- ...

I'm relatively new to modern web dev and find the above a bit overwhelming, e.g. I'm embarrassed to share it took me ~3 hours the other day to create and configure a supabase with a vercel app and resolve a few errors. The second you stray just slightly from the "getting started" tutorial in the docs you're suddenly in the wilderness. It's not even code, it's... configurations, plumbing, orchestration, workflows, best practices. A lot of glory will go to whoever figures out how to make it accessible and "just work" out of the box, for both humans and, increasingly and especially, AIs.

12:17 AM · Mar 27, 2025 · **1.7M** Views

# Hands-on: Building a Quiz App

# Q&A & Break

# Spec Doc Template

## # Quiz Application Specification

### ## Problem Statement

Create a simple quiz application that allows users to test their knowledge on vari

### ## Core Features

1. Present questions with multiple choice answers
2. Track user score
3. Show results at the end
4. Save progress locally

### ## Technical Requirements

- HTML/CSS/JavaScript only
- Mobile-responsive design
- Local storage for saving progress
- No external dependencies

# Cursor Project Rules Files

Cursor Project Rules: consistency and context for AI-generated code:

```
# `./.cursor/rules`
```

## ## Project Context

- If you need information regarding the UI, refer to *UI\_Development\_Plan.md*
- For database queries, refer to *DB\_Design.md*
- For task progress, check *Project\_Status.md*
- Always reference @Docs for official documentation

## ## Coding Standards

- Use camelCase for variables and functions
- Use PascalCase for component names
- Add JSDoc comments for all functions
- Follow accessibility best practices

## ## Project Structure

- Keep components in separate files
- Store utility functions in *utils.js*
- Use consistent error handling
- Update *Project\_Status.md* at end of sessions

## ## AI Assistance Guidelines

- Break tasks into smallest possible steps
- Reference context files for each task
- Use design inspiration for UI refinement
- Always review and test generated code

# Hands-on: Putting Everything Together

# Q&A & Break

# Extending capabilities with APIs

# Extending with APIs: The Basics

What is an API?

# Extending with APIs: The Basics

## What is an API?

- Application Programming Interface - a way for different software to communicate

# Extending with APIs: The Basics

## What is an API?

- Application Programming Interface - a way for different software to communicate

## Why use APIs?

- They let your app access external services and data

# Extending with APIs: The Basics

Types of APIs for beginners:

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)
- Simple databases (Supabase, Firebase)

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)
- Simple databases (Supabase, Firebase)
- Image generation (OpenAI DALL-E)

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)
- Simple databases (Supabase, Firebase)
- Image generation (OpenAI DALL-E)

## API Integration Process:

1. Get an API key (usually free for low usage)

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)
- Simple databases (Supabase, Firebase)
- Image generation (OpenAI DALL-E)

## API Integration Process:

1. Get an API key (usually free for low usage)
2. Make requests using JavaScript's `fetch()` function

# Extending with APIs: The Basics

## Types of APIs for beginners:

- Weather data (OpenWeatherMap)
- Maps and location (Google Maps)
- Simple databases (Supabase, Firebase)
- Image generation (OpenAI DALL-E)

## API Integration Process:

1. Get an API key (usually free for low usage)
2. Make requests using JavaScript's `fetch()` function
3. Process and display the returned data

# Hands-on: Interactive Live Coding

# No-Code/Low-Code AI Tools

- **Lovable**

- Create full stack apps by chatting with AI
- DB integration
- Prototype more complex functionality fast

- **Vercel V0**

- Text-to-interface generation
- React component creation
- Integrated with Next.js

- **Replit**

- Collaborative coding environment
- Built-in hosting
- AI coding assistant (Ghostwriter)

- **Replicate**

- Model deployment
- API access to various AI models
- Custom model hosting

# Connect With Me



[Course materials](#)



[LinkedIn](#)



[Twitter/X](#)



[YouTube](#)



Email: [lucasenkrateia@gmail.com](mailto:lucasenkrateia@gmail.com)

