DESIGNED CHANGE

Building a Production Web Application with AI

From First Prompt to Shipped Product in 9 Days

Sean Hobson
Chief Design Officer, EdPlus at Arizona State University
Clinical Assistant Professor, Arizona State University

February 2026

Built entirely with Claude (Anthropic) by a non-developer.

4,500+	**45**	**9**	**$0**
Lines of Code	React Components	Days to Launch	Developer Cost
27	**5**	**$200**	**$25-38K**
Assessment Questions	Dimensions	Total Build Cost	Est. Replacement

Executive Summary

On February 16, 2026, I bought the domain designedchange.ai and set out to build a web-based strategic assessment tool grounded in my doctoral research on Learning Innovation Departments and transformational change in higher education. I had a clear vision for what the tool should do, a deep understanding of the content, and zero coding experience.

Nine days later, designedchange.ai was live in production with real users. It is a full-stack web application with 27 research-backed assessment questions across 5 strategic dimensions, an AI-powered research agent (powered by the Claude API), automated email delivery, a Supabase database capturing every response, an admin dashboard, and SEO infrastructure. Users receive immediate, personalized results with dimension-level scoring, peer benchmarks, and actionable recommendations.

The entire application was built through conversation with Claude, Anthropic's AI assistant. Every line of code was generated through natural language prompts. My role was to provide the domain expertise, make design decisions, and test the output. Claude wrote the code, debugged errors, and suggested architecture decisions I would not have known to make.

This document walks through the entire process, step by step, for anyone interested in building their own AI-assisted project.

What Was Built

Designed Change (designedchange.ai) is a strategic readiness assessment for learning organizations. It measures how intentionally an organization has designed its approach to change across five dimensions drawn from my doctoral research at Dublin City University and published work with Johns Hopkins University Press.

Core Features

- **27-question adaptive assessment** across 5 research-backed dimensions (Aspirations, Positioning, Architecture, Capabilities, Culture)
- **Dual scoring throughlines** (Design vs. Strategy) showing where the organization is intentional vs. reactive
- **AI-powered research agent** using the Claude API, grounded in the scholarship behind the tool
- **Automated email delivery** via Resend, sending branded HTML results reports and admin notifications
- **Supabase backend** capturing assessments, AI conversations, and email requests across 3 database tables
- **Admin dashboard** with KPI cards, dimension analytics, conversation viewer, and CSV export
- **Personalized results page** with readiness level, dimension breakdowns, gap analysis, and recommendations

- **SEO infrastructure** including structured data, sitemap, robots.txt, Open Graph tags, and llms.txt
- **Microsoft Clarity analytics** tracking user behavior, session recordings, and heatmaps
- **Progressive UX enhancements** including progress tracking, social proof, browser exit warnings, and localStorage persistence

Technology Stack

Layer	Technology
Frontend	React 18 (single-file JSX), Vite build tool
Backend / API	Vercel Serverless Functions (Node.js)
Database	Supabase (PostgreSQL) with Row Level Security
AI Agent	Claude API (Anthropic) via serverless proxy
Email	Resend API with branded HTML templates
Hosting	Vercel (auto-deploy from GitHub)
Analytics	Microsoft Clarity (session recordings, heatmaps)
Domain	Namecheap (designedchange.ai)
Version Control	GitHub
Build Tool	Claude (Anthropic) — all code AI-generated

The Build Process: Step by Step

What follows is the actual sequence of how this application was built. Each step represents a conversation (or series of conversations) with Claude, followed by me copying the generated code into GitHub, which auto-deployed to production via Vercel.

STEP 1

Vision and Architecture

I described what I wanted to build in plain English. Claude proposed the technology stack and created the initial project structure.

> *My prompt (paraphrased): I want to build a web-based assessment tool based on my doctoral research. It should ask ~27 questions across 5 dimensions, score users, and give them personalized results. I want it to look elegant and minimal.*
>
> *Claude delivered: Complete React application in a single JSX file, Vite configuration, package.json, and deployment instructions for Vercel.*
>
> *Key decision: Single-file architecture (one App.jsx) for simplicity — no complex multi-file project for a solo non-developer to manage.*

STEP 2

Content and Assessment Design

I provided all 27 questions organized by dimension, the scoring rubric, readiness levels, and the research framework. Claude wired it into the interactive assessment flow.

> *Input: My research framework (Playing to Win adapted for learning orgs), 27 Likert-scale questions with Design/Strategy axis tags, 5 readiness levels.*
>
> *Claude delivered: Full assessment engine with dimension tracking, dual-axis scoring, progress persistence, and animated transitions.*
>
> *Key decision: Dual throughlines (Design vs. Strategy) as a differentiator.*

STEP 3

Results and Visualization

I described the results experience I wanted. Claude built the full results dashboard with dimension breakdowns, peer benchmarks, gap analysis, and personalized recommendations.

> *~4 rounds of refinement on layout, color, copy, and the recommendation logic.*

STEP 4

AI Research Agent

I wanted users to be able to ask questions about their results and the research. Claude built a conversational AI agent powered by its own API.

Architecture: System prompt with full research knowledge base → Vercel serverless function proxying to Claude API → floating chat panel in the React app.

Key feature: Context-aware — the agent knows the user's scores, readiness level, and weakest dimensions.

Iteration: Initially too narrow (only discussed my research). Updated system prompt to draw on broader field knowledge.

STEP 5

Database and Data Capture

I set up a Supabase project (free tier) and Claude built all the data capture logic to store every assessment, conversation, and email request.

Three tables: assessments (full response data, scores, demographics), conversations (AI chat threads), email_requests (delivery log).

Key debug: Data wasn't saving — turned out to be a missing column in the database schema.

STEP 6

Email System

Users receive branded HTML results emails. I get admin notifications on every completion. Built with Resend API.

Two serverless functions for user reports and admin alerts.

Key learning: New sending domains go to junk initially. Need DMARC/SPF/DKIM records.

STEP 7

Design Polish and UX

Multiple rounds of visual refinement. Warm minimalism inspired by editorial design. Georgia serif for headings, clean sans-serif for body. Amber accent on stone/warm gray palette.

UX enhancements (from Clarity data): Browser exit warning, localStorage persistence, social proof counter, time estimates per dimension.

STEP 8

Admin Dashboard

A standalone HTML dashboard for monitoring all data from Supabase — without deploying to production.

Features: Tabs for Assessments, Conversations, Emails. KPI cards, dimension averages, readiness distribution, CSV export.

STEP 9

SEO and Discoverability

Full audit of search optimization. Structured data, sitemap, meta tags, Google Search Console setup.

Person schema (JSON-LD), keyword-optimized title, Open Graph image, sitemap, robots.txt, llms.txt for AI crawlers.

Cost Breakdown

The total out-of-pocket cost to build and run this application is remarkably low compared to traditional development approaches.

Build Costs

Item	Cost	Notes
Claude Max subscription	$100/mo	Primary build tool
Anthropic API credits	~$5-10	Per-query cost for live agent
Vercel hosting	$0	Free tier, auto-deploy
Supabase database	$0	Free tier, 500MB
Resend email	$0	Free tier, 100 emails/day
Namecheap domain	~$12/yr	designedchange.ai
Microsoft Clarity	$0	Session recordings, heatmaps
GitHub	$0	Version control

Total build cost: approximately $200 (one month of Claude Max + domain + minor API usage). Estimated replacement cost if hiring a developer or agency: **$25,000–38,000**.

Ongoing Monthly Costs

- **Claude Max:** $100/month (continued development and iteration)
- **Anthropic API:** ~$5–15/month (scales with chatbot usage)
- **All other services:** $0 on free tiers at current scale

Total ongoing cost is approximately $100–120/month.

What I Learned

AI as a Force Multiplier

The most important insight is that AI doesn't replace expertise — it amplifies it. I brought deep domain knowledge about organizational change, learning innovation, and strategic assessment design. Claude brought the ability to turn that knowledge into working software. Neither could have done this alone.

The pattern was consistent: I described what I wanted in terms of user experience and strategic intent. Claude translated that into code. When something didn't work, I described the problem (often with screenshots), and Claude diagnosed and fixed it. The feedback loop was measured in minutes, not days.

Speed and Cost Savings

What would have taken a traditional development team 2–3 months and $25,000+ was accomplished in 9 days for under $200. More importantly, the iteration speed meant I could test ideas immediately. If a design choice didn't feel right, I could describe the change and have a new version deployed in under an hour.

Innovation in Practice

There is a certain irony in using AI to build a tool about designed change. The process of building Designed Change was itself an act of designed change — intentionally choosing a new approach, iterating on the design, and shipping something that would not have been possible with traditional methods. I used the very framework I was researching.

Learning the Tools

I started this project with no knowledge of React, JavaScript, Vercel, Supabase, or API development. I still don't know how to write code. What I learned was how to describe what I want clearly, how to debug by describing symptoms, and how to manage a software project through conversation. These are transferable skills that will apply to every future project.

What Went Wrong

Not everything was smooth. Environment variables that seemed configured weren't available at build time. A missing database column caused silent failures. An import error crashed the live site for a few minutes. Each of these required patient debugging — and taught me that software development is as much about diagnosis as it is about creation.

How to Do This Yourself

If you have deep expertise in a subject and a clear idea of what you want to build, here is the playbook:

- **Start with the outcome, not the technology.** Describe what your users should experience. Let the AI recommend the architecture.
- **Work in one file as long as possible.** Complexity is the enemy of a solo non-developer.
- **Deploy early, iterate constantly.** We had a live site within the first session. Every change after that was incremental improvement.
- **Use free tiers aggressively.** Vercel, Supabase, Resend, GitHub, and Clarity are all free at startup scale.
- **Describe problems, not solutions.** When something breaks, describe what you see. Don't try to guess the technical fix.
- **Build the dashboard early.** Being able to see your data gives you confidence and reveals insights about user behavior.

• **Let analytics drive iteration.** Microsoft Clarity showed us exactly where users dropped off. That data directly informed our UX enhancements.

Designed Change is live and collecting data. The bigger story is what this represents for professionals with domain expertise who want to build tools, products, and platforms. The barrier to entry for software development has fundamentally changed. You don't need to learn to code. You need to learn to describe what you want with clarity and precision — and then iterate.

Try the assessment: **designedchange.ai**

Questions? sean.hobson@asu.edu

DESIGNED CHANGE

Building a Production Web Application with AI

From First Prompt to Shipped Product in 9 Days

Sean Hobson
Chief Design Officer, EdPlus at Arizona State University
Clinical Assistant Professor, Arizona State University

February 2026

Built entirely with Claude (Anthropic) by a non-developer.

4,500+	**45**	**9**	**$0**
Lines of Code	React Components	Days to Launch	Developer Cost
27	**5**	**$200**	**$25-38K**
Assessment Questions	Dimensions	Total Build Cost	Est. Replacement

Executive Summary

On February 16, 2026, I bought the domain designedchange.ai and set out to build a web-based strategic assessment tool grounded in my doctoral research on Learning Innovation Departments and transformational change in higher education. I had a clear vision for what the tool should do, a deep understanding of the content, and zero coding experience.

Nine days later, designedchange.ai was live in production with real users. It is a full-stack web application with 27 research-backed assessment questions across 5 strategic dimensions, an AI-powered research agent (powered by the Claude API), automated email delivery, a Supabase database capturing every response, an admin dashboard, and SEO infrastructure. Users receive immediate, personalized results with dimension-level scoring, peer benchmarks, and actionable recommendations.

The entire application was built through conversation with Claude, Anthropic's AI assistant. Every line of code was generated through natural language prompts. My role was to provide the domain expertise, make design decisions, and test the output. Claude wrote the code, debugged errors, and suggested architecture decisions I would not have known to make.

This document walks through the entire process, step by step, for anyone interested in building their own AI-assisted project.

What Was Built

Designed Change (designedchange.ai) is a strategic readiness assessment for learning organizations. It measures how intentionally an organization has designed its approach to change across five dimensions drawn from my doctoral research at Dublin City University and published work with Johns Hopkins University Press.

Core Features

- **27-question adaptive assessment** across 5 research-backed dimensions (Aspirations, Positioning, Architecture, Capabilities, Culture)
- **Dual scoring throughlines** (Design vs. Strategy) showing where the organization is intentional vs. reactive
- **AI-powered research agent** using the Claude API, grounded in the scholarship behind the tool
- **Automated email delivery** via Resend, sending branded HTML results reports and admin notifications
- **Supabase backend** capturing assessments, AI conversations, and email requests across 3 database tables
- **Admin dashboard** with KPI cards, dimension analytics, conversation viewer, and CSV export
- **Personalized results page** with readiness level, dimension breakdowns, gap analysis, and recommendations

- **SEO infrastructure** including structured data, sitemap, robots.txt, Open Graph tags, and llms.txt
- **Microsoft Clarity analytics** tracking user behavior, session recordings, and heatmaps
- **Progressive UX enhancements** including progress tracking, social proof, browser exit warnings, and localStorage persistence

Technology Stack

Layer	Technology
Frontend	React 18 (single-file JSX), Vite build tool
Backend / API	Vercel Serverless Functions (Node.js)
Database	Supabase (PostgreSQL) with Row Level Security
AI Agent	Claude API (Anthropic) via serverless proxy
Email	Resend API with branded HTML templates
Hosting	Vercel (auto-deploy from GitHub)
Analytics	Microsoft Clarity (session recordings, heatmaps)
Domain	Namecheap (designedchange.ai)
Version Control	GitHub
Build Tool	Claude (Anthropic) — all code AI-generated

The Build Process: Step by Step

What follows is the actual sequence of how this application was built. Each step represents a conversation (or series of conversations) with Claude, followed by me copying the generated code into GitHub, which auto-deployed to production via Vercel.

STEP 1

Vision and Architecture

I described what I wanted to build in plain English. Claude proposed the technology stack and created the initial project structure.

My prompt (paraphrased): I want to build a web-based assessment tool based on my doctoral research. It should ask ~27 questions across 5 dimensions, score users, and give them personalized results. I want it to look elegant and minimal.

Claude delivered: Complete React application in a single JSX file, Vite configuration, package.json, and deployment instructions for Vercel.

Key decision: Single-file architecture (one App.jsx) for simplicity — no complex multi-file project for a solo non-developer to manage.

STEP 2

Content and Assessment Design

I provided all 27 questions organized by dimension, the scoring rubric, readiness levels, and the research framework. Claude wired it into the interactive assessment flow.

Input: My research framework (Playing to Win adapted for learning orgs), 27 Likert-scale questions with Design/Strategy axis tags, 5 readiness levels.

Claude delivered: Full assessment engine with dimension tracking, dual-axis scoring, progress persistence, and animated transitions.

Key decision: Dual throughlines (Design vs. Strategy) as a differentiator.

STEP 3

Results and Visualization

I described the results experience I wanted. Claude built the full results dashboard with dimension breakdowns, peer benchmarks, gap analysis, and personalized recommendations.

~4 rounds of refinement on layout, color, copy, and the recommendation logic.

STEP 4

AI Research Agent

I wanted users to be able to ask questions about their results and the research. Claude built a conversational AI agent powered by its own API.

Architecture: System prompt with full research knowledge base → Vercel serverless function proxying to Claude API → floating chat panel in the React app.

Key feature: Context-aware — the agent knows the user's scores, readiness level, and weakest dimensions.

Iteration: Initially too narrow (only discussed my research). Updated system prompt to draw on broader field knowledge.

STEP 5

Database and Data Capture

I set up a Supabase project (free tier) and Claude built all the data capture logic to store every assessment, conversation, and email request.

Three tables: assessments (full response data, scores, demographics), conversations (AI chat threads), email_requests (delivery log).

Key debug: Data wasn't saving — turned out to be a missing column in the database schema.

STEP 6

Email System

Users receive branded HTML results emails. I get admin notifications on every completion. Built with Resend API.

Two serverless functions for user reports and admin alerts.

Key learning: New sending domains go to junk initially. Need DMARC/SPF/DKIM records.

STEP 7

Design Polish and UX

Multiple rounds of visual refinement. Warm minimalism inspired by editorial design. Georgia serif for headings, clean sans-serif for body. Amber accent on stone/warm gray palette.

UX enhancements (from Clarity data): Browser exit warning, localStorage persistence, social proof counter, time estimates per dimension.

STEP 8

Admin Dashboard

A standalone HTML dashboard for monitoring all data from Supabase — without deploying to production.

Features: Tabs for Assessments, Conversations, Emails. KPI cards, dimension averages, readiness distribution, CSV export.

STEP 9

SEO and Discoverability

Full audit of search optimization. Structured data, sitemap, meta tags, Google Search Console setup.

Person schema (JSON-LD), keyword-optimized title, Open Graph image, sitemap, robots.txt, llms.txt for AI crawlers.

Cost Breakdown

The total out-of-pocket cost to build and run this application is remarkably low compared to traditional development approaches.

Build Costs

Item	Cost	Notes
Claude Max subscription	$100/mo	Primary build tool
Anthropic API credits	~$5-10	Per-query cost for live agent
Vercel hosting	$0	Free tier, auto-deploy
Supabase database	$0	Free tier, 500MB
Resend email	$0	Free tier, 100 emails/day
Namecheap domain	~$12/yr	designedchange.ai
Microsoft Clarity	$0	Session recordings, heatmaps
GitHub	$0	Version control

Total build cost: approximately $200 (one month of Claude Max + domain + minor API usage). Estimated replacement cost if hiring a developer or agency: **$25,000–38,000**.

Ongoing Monthly Costs

- **Claude Max:** $100/month (continued development and iteration)
- **Anthropic API:** ~$5–15/month (scales with chatbot usage)
- **All other services:** $0 on free tiers at current scale

Total ongoing cost is approximately $100–120/month.

What I Learned

AI as a Force Multiplier

The most important insight is that AI doesn't replace expertise — it amplifies it. I brought deep domain knowledge about organizational change, learning innovation, and strategic assessment design. Claude brought the ability to turn that knowledge into working software. Neither could have done this alone.

The pattern was consistent: I described what I wanted in terms of user experience and strategic intent. Claude translated that into code. When something didn't work, I described the problem (often with screenshots), and Claude diagnosed and fixed it. The feedback loop was measured in minutes, not days.

Speed and Cost Savings

What would have taken a traditional development team 2–3 months and $25,000+ was accomplished in 9 days for under $200. More importantly, the iteration speed meant I could test ideas immediately. If a design choice didn't feel right, I could describe the change and have a new version deployed in under an hour.

Innovation in Practice

There is a certain irony in using AI to build a tool about designed change. The process of building Designed Change was itself an act of designed change — intentionally choosing a new approach, iterating on the design, and shipping something that would not have been possible with traditional methods. I used the very framework I was researching.

Learning the Tools

I started this project with no knowledge of React, JavaScript, Vercel, Supabase, or API development. I still don't know how to write code. What I learned was how to describe what I want clearly, how to debug by describing symptoms, and how to manage a software project through conversation. These are transferable skills that will apply to every future project.

What Went Wrong

Not everything was smooth. Environment variables that seemed configured weren't available at build time. A missing database column caused silent failures. An import error crashed the live site for a few minutes. Each of these required patient debugging — and taught me that software development is as much about diagnosis as it is about creation.

How to Do This Yourself

If you have deep expertise in a subject and a clear idea of what you want to build, here is the playbook:

- **Start with the outcome, not the technology.** Describe what your users should experience. Let the AI recommend the architecture.
- **Work in one file as long as possible.** Complexity is the enemy of a solo non-developer.
- **Deploy early, iterate constantly.** We had a live site within the first session. Every change after that was incremental improvement.
- **Use free tiers aggressively.** Vercel, Supabase, Resend, GitHub, and Clarity are all free at startup scale.
- **Describe problems, not solutions.** When something breaks, describe what you see. Don't try to guess the technical fix.
- **Build the dashboard early.** Being able to see your data gives you confidence and reveals insights about user behavior.

• **Let analytics drive iteration.** Microsoft Clarity showed us exactly where users dropped off. That data directly informed our UX enhancements.

Designed Change is live and collecting data. The bigger story is what this represents for professionals with domain expertise who want to build tools, products, and platforms. The barrier to entry for software development has fundamentally changed. You don't need to learn to code. You need to learn to describe what you want with clarity and precision — and then iterate.

Try the assessment: **designedchange.ai**

Questions? sean.hobson@asu.edu