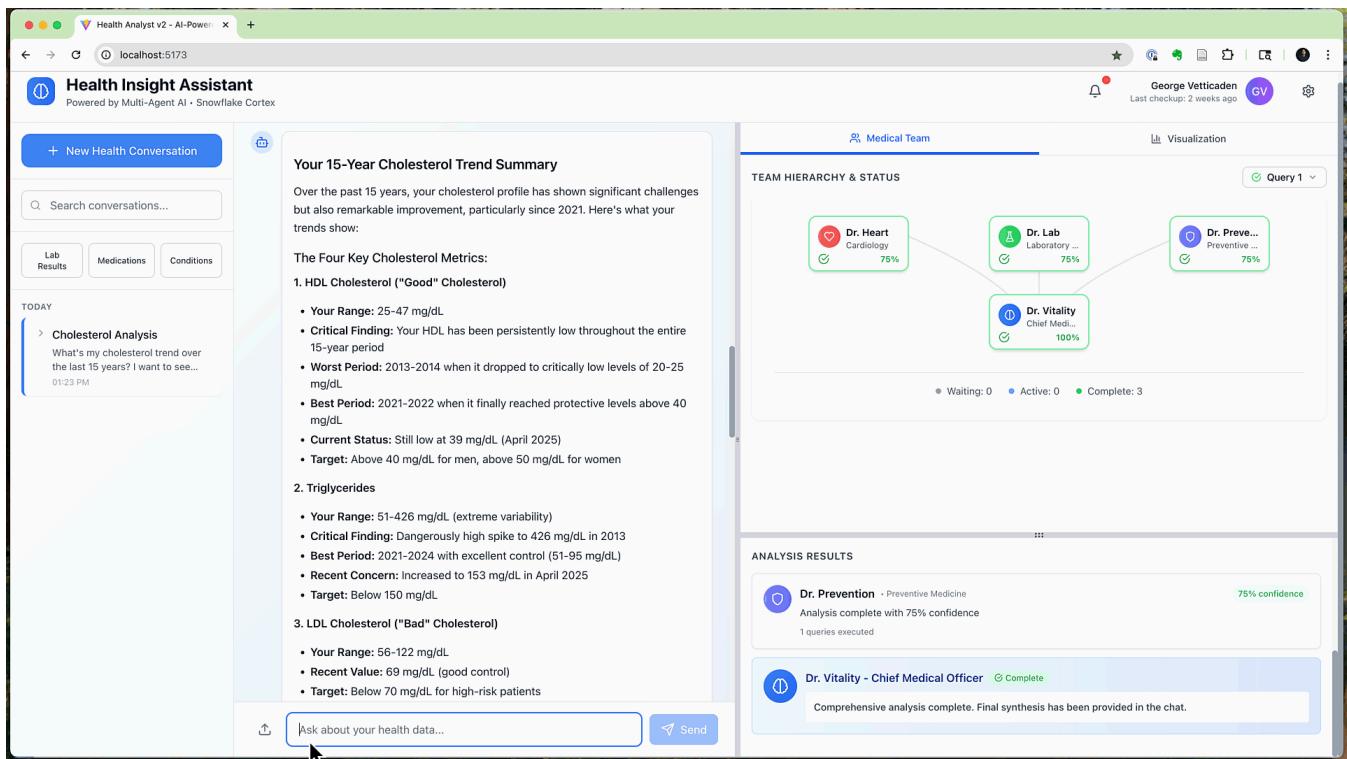


The 3 Amigo Agents: The Claude Code Development Pattern I Discovered While Implementing Anthropic's Multi-Agent Architecture

My health insights system had one AI agent handling cardiology, endocrinology, pharmacy, and general practice simultaneously. Anthropic's research showed me an orchestrated multi-agent architecture. However, rebuilding with Dr. Heart, Dr. Pharma, and Dr. Hormone led to a bigger discovery: the 3 Amigo Agents pattern, which now powers all my AI development.



The 3 Amigo Agents Build Multi-Agent Systems: Using PM Agent, UX Designer Agent, and Claude Code to develop Anthropic's multi-agent orchestration pattern — creating sophisticated medical team interfaces where AI specialists collaborate in real-time

One Health Analyst Agent, 15 Medical Domains: From Solo Practitioner to Medical Team

When Anthropic published "[How we built our multi-agent research system](#)", I immediately recognized the architectural constraints I'd been facing with my health insights system — a single AI agent attempting to analyze 200+ pages of Apple Health data across 15 medical domains. Their key finding validated what I'd been experiencing:

“We found that a multi-agent system with Claude Opus 4 as the lead agent and Claude Sonnet 4 subagents outperformed single-agent Claude Opus 4 by 90.2% on our internal research eval.”

90.2% performance improvement using multi-agent orchestration. Anthropic found that when problems become too complex, single agents hit a wall. Their solution: an orchestrator/worker pattern where a lead researcher coordinates specialized agents working in parallel, similar to a chief medical officer managing medical specialists.

Reading this crystallized two priorities. First, refactor my health insights system to harness this multi-agent architecture. Second, create a proper eval framework to validate the approach.

Building the health system led to an unexpected discovery. I found myself orchestrating three specialized agents — Product Manager Agent for requirements, UX Designer Agent for design, and Claude Code for implementation. This “3 Amigo Agents” pattern emerged naturally but transformed my workflow, reducing development from weeks to hours.

The 3 Amigo Agents Pattern in Action: Complete technical demonstration showing PM Agent creating specifications, UX Agent designing the medical team interface, and Claude Code implementing the full multi-agent health system — transforming empty directories into a working application with 8 specialized medical agents in under 3 hours

I'll share these learnings in two parts. This first part reveals the 3 Amigo Agents pattern — the development approach I discovered that transforms weeks of AI development into

hours. In Part 2, I'll dive deep into the multi-agent health system that sparked this discovery, including the comprehensive evaluation framework that validated Anthropic's 90.2% performance gains.

The Gap: Why I Needed More Than Claude's Consumer Tools

Implementing Anthropic's multi-agent pattern meant building something far more complex than my original health system. While Claude.ai and Claude Desktop had served me well for that first version, this new architecture demanded capabilities they simply couldn't provide:

- **Custom orchestration logic** using Anthropic's APIs directly for parallel agent execution
- **Thread management system** with UUID-based conversation tracking and full history persistence
- **Professional UI with real-time streaming** from 8+ medical specialists simultaneously
- **Dynamic visualization engine** generating health-specific React components on demand
- **Sophisticated error handling** with retry logic and graceful degradation
- **Evaluation framework** to validate the 90.2% performance improvements

The requirements read like a product roadmap for a venture-backed startup: 25+ custom UI components, glassmorphism design language, three-panel responsive layout, medical team visualization with real-time status updates, query-based visualization history, SSE streaming, accessibility compliance... the list went on.

This wasn't a weekend project anymore. This was enterprise-grade software that would typically require a team of developers and months of work. The challenge was clear: how could a small team tackle something this complex without months of architectural planning and implementation work?

The 3 Amigo Agents: My Secret Weapon for Rapid Development

The solution emerged from an unexpected source: the very process of trying to build the system. As I wrestled with requirements, design, and implementation, I found myself naturally gravitating toward three specialized AI agents — each handling what a traditional team member would do.

I call this pattern the “3 Amigo Agents” — a modern evolution of the traditional product development triad where Product Manager, UX Design Lead, and Engineering Lead collaborate closely.

In my AI-powered version, three specialized agents work together with me at the center:

- **Product Manager Agent:** Transforms vision into comprehensive requirements, PRDs, and technical architecture
- **UX Designer Agent:** Creates interactive prototypes, design systems, and component specifications
- **Claude Code:** Takes all requirements and designs to implement the complete solution

I sit at the center of this triad, orchestrating collaboration between three specialized agents that mirror these traditional roles.

Meet the 3 Amigo Agents: The Future of Product Development

One Product Owner, Three AI Specialists, Infinite Possibilities

Your Complete AI Product Team Working in Perfect Harmony



Orchestrated Workflow: Product Owner orchestrates a continuous development cycle where requirements flow from PM Agent to Designer Agent to Claude Code, transforming ideas into reality in perfect harmony

The 3 Amigos in Action: How the Pattern Works

What makes this pattern powerful is how each agent builds on the previous one's work. The PM Agent doesn't just list features — it defines the core architecture. The UX Agent doesn't just make things pretty — it translates architecture into experiential design. And Claude Code doesn't just implement — it brings both vision and design to life with everything it needs to succeed.

This orchestrated collaboration creates rich, multi-layered context that transforms Claude Code from a coding assistant into a builder that delivers working MVPs with just a few iterations.

The 3 Amigos Pipeline: From Vision to Implementation

From initial vision to working system—how three AI agents collaborate to build enterprise software



The 3 Amigos Pipeline: Your 5 initial documents flow through PM Agent (creating 7 artifacts), then UX Agent (adding 8 more), culminating in 20+ comprehensive artifacts that enable Claude Code to build complex, functional systems with minimal iterations.

This isn't just document multiplication — it's progressive refinement. Each agent adds its specialized lens: the PM Agent transforms ideas into structured requirements, the UX Agent transforms requirements into visual experiences, and Claude Code transforms everything into working software. The CLAUDE.md file ensures nothing gets lost in translation.

Let me show you what this looks like in practice.

The PM Agent: From Vision to Product Strategy in 20 Minutes

Having spent the last decade as a product leader — preceded by roles as engineering leader, technical architect, and field CTO — I know firsthand that product management

is one of tech's hardest jobs. You're the CEO without authority, responsible for business outcomes while navigating engineering constraints, stakeholder politics, and market dynamics.

The PM Agent faces a similar monumental challenge: transforming a product owner's vision into comprehensive specifications that balance user needs, technical feasibility, and business constraints.

Through my experience, I've learned that great PMs can only deliver exceptional results when they have deep context — the kind that comes from immersing yourself in customer data, understanding technical constraints, knowing the competitive landscape, and aligning with business objectives. The PM Agent is no different. To unlock its full potential, I needed to provide the same comprehensive context I'd give to a senior PM joining my team.

This realization led me to develop a strategic framework that mirrors how I onboard new PMs in the real world:



Strategic Framework: The three pillars of comprehensive PM onboarding — domain requirements & constraints, best practices & proven patterns, and expected deliverables with success criteria

This framework mirrors exactly how I'd onboard a senior PM to own a complex product:

- **Domain expertise & constraints** that ground the product in real user needs, technical realities, and business context
- **Proven patterns & best practices** (like Anthropic's multi-agent architecture) that provide blueprints for success
- **Clear deliverables & expectations** that define exactly what stakeholders, designers, and engineers will need

Just as I wouldn't expect a new PM to succeed with "build a health app," I couldn't expect the PM Agent to deliver professional-grade specifications without equivalent context. Let me show you what this comprehensive onboarding looks like in practice:

Product Management Agent

Private



...

An AI Product Manager that transforms domain requirements into comprehensive product specifications. Creates PRDs, user stories, technical architecture, API specs, and data models for multi-agent systems. Specializes in orchestrator-worker patterns and follows Anthropic's best practices for building production-ready AI applications.

Collapse description

I need you to create a comprehensive product specification for a production-ready Multi-Agent Health Insight System. This system should demonstrate enterprise-grade features including conversation persistence, visualization history, and comprehensive error handling.

Core Requirements:

1. **Multi-Agent Architecture**

- Design a Chief Medical Officer (CMO) as the orchestrator agent
- Include 8 medical specialist agents (Cardiology, Endocrinology, Lab Medicine, Data Analytics, Preventive Medicine, Pharmacy, Nutrition, Primary Care)
- Implement progressive disclosure of health analysis results
- Include a visualization agent that generates health-specific React components

2. **Thread Management System**



Research



Claude Opus 4



health-technical-customization-guide.md

239 lines



multi-agent-implementation-architecture.md

207 lines



simplified-architecture-brief.md

103 lines



tool-interface-document.md

246 lines



health-domain-requirements.md

127 lines



multi-archit brief.r

66 lines



Initial Prompt: A comprehensive PM onboarding package — 7 detailed documents plus Anthropic's blog, totaling over 50 pages of context. Notice how I provide the same depth of information I'd give a senior PM joining my team.

Twenty minutes later, here's what came back:

I've created all 8 comprehensive product artifacts for your Multi-Agent Health Insight System. Here's what I've delivered:

Delivered Artifacts

- 1. Product Requirements Document (PRD.md)**
 - Complete specification of the multi-agent health system
 - Core features: thread management, 8 medical specialists, visualization history
 - Production requirements: error handling, performance metrics, HIPAA considerations
 - Success criteria and risk mitigation strategies
- 2. User Stories (user-stories.md)**
 - 20+ detailed user stories with acceptance criteria
 - Covers: health consultations, real-time analysis, visualization history, error recovery
 - Includes mobile experience and accessibility stories
 - Production-ready scenarios like network interruption recovery
- 3. API Specification (api-specification.md)**
 - Complete REST + SSE endpoint documentation
 - Thread management, health analysis streaming, visualization endpoints
 - Detailed SSE event types for real-time medical team updates
 - Export functionality for physician reports
- 4. Data Models (data-models.md)**
 - Comprehensive TypeScript interfaces for all entities
 - Health-specific models: HealthMetric, Medication, SpecialistAgent
 - Thread persistence schemas with UUID tracking
 - Error handling and audit models
- 5. Component Architecture (component-architecture.md)**
 - 45+ React components organized by domain
 - Medical team visualization components with real-time status
 - Health-specific components (LabResultTable, RiskGauge, MedicationCard)
 - Performance optimizations and accessibility components
- 6. Tool Interface Documentation (tool-interface.md)**
 - Detailed guide for pre-built Snowflake health tools
 - Clear examples of health queries and expected responses
 - Integration patterns for CMO and specialist agents
 - Critical reminder: DO NOT reimplement these tools

Reply to Claude... + Research

Claude Opus 4 ↑

Switch between artifacts

- Feature Priority Document - Health Insight System Document
- System Architecture - Health Insight System Document
- Tool Interface Documentation - Health Insight System Document
- Component Architecture - Health Insight System Document
- Data Models - Health Insight System Document
- API Specification - Health Insight System Document

1. Product 2. User 3. Comp 4. Techn Resource

MVP Time

Growth 3

Conclusion

This project and build is adjusted to

PM Agent Complete Deliverables: 8 comprehensive documents including PRD, user stories with acceptance criteria, system architecture, API specifications, data models, component architecture, tool interface documentation, and feature priorities — over 100 pages of product specifications delivered in 20 minutes.

The PM Agent absorbed my seven input documents, plus Anthropic's blog link — over 50 pages of context — and delivered eight comprehensive product artifacts in just 20 minutes: PRD, user stories, system architecture, API specs, data models, component architecture, tool interfaces, and prioritized features. The kind of comprehensive product documentation that typically takes **2–3 weeks** of iteration.

But specifications alone don't create great products — we needed it to come alive visually.

The UX Agent: From Product Specifications to Visual Experience

Armed with the PM Agent's 8 comprehensive product specifications plus my 4 original design documents, I approached the UX Designer Agent with a rich context package — 12

documents that would guide the visual transformation of these requirements into a healthcare interface.

UX Designer Agent

Private

An AI UX Designer that transforms product requirements into beautiful, functional interfaces. Creates design systems, component specifications, interactive prototypes, and visual mockups. Specializes in modern UI patterns, accessibility, and crafting intuitive user experiences for complex multi-agent AI systems.

Collapse description

I need you to design a production-ready, modern UI/UX for a Multi-Agent Health Insight System that feels as polished as a manually crafted medical application. The design should emphasize ~~glassmorphism~~, smooth animations, and sophisticated health information architecture.

Critical Design Requirements:

1. **Component Library (25+ Components)**
You MUST design and specify these health-focused components with variants:
Layout Components:
 - MainLayout (three-panel responsive health dashboard)
 - Header (Health Insight branding, user profile, settings)
 - ThreadSidebar (health consultation history)
 - ResizablePanel (draggable dividers)

+ Research Claude Opus 4 ↑

PRD.md 209 lines MD	user-stories.md 273 lines MD	api-specification.md 540 lines MD	data-models.md 608 lines MD	component-architecture.md 653 lines MD	tool-ir 511 lines MD
---------------------------	------------------------------------	---	-----------------------------------	--	----------------------------

Design Brief: 12 comprehensive documents (8 from PM Agent + 4 from Product Owner) create unprecedented context for the UX Agent

The success came from structuring the UX prompt around four design pillars that would create a consumer-facing healthcare interface where the multi-agent pattern manifests as a medical team:

UX Designer Agent: Trust Through Design

Transforming medical requirements into interfaces that inspire confidence



Glassmorphism
Medical Trust



Components
Health Toolkit



Animations
Calming Motion



Team Viz
8 Specialists



3-Panel Layout
Threads • Chat • Visuals



Medical Colors
Trust & Clarity



Depth Layers
Focus & Hierarchy



Responsive
Any Device

Medical Requirements + Design Excellence = Trusted Health Interface

Strategic Design Framework: Four pillars for consumer healthcare — glassmorphism for trust, 25+ components for comprehensive medical UI, animations for real-time health updates, and medical team visualization showing CMO orchestrating 8 specialists

In 25 minutes, the UX Agent transformed these requirements into a complete design system that would typically take a design team **weeks** to deliver:

Completed design documentation for Multi-Agent Health Insight System.

ts

Design Package Complete!

I've created a comprehensive design system for your Multi-Agent Health Insight System with the following deliverables:

1. Design System ([health-design-system.md](#))

- Medical-grade color palette with exact specialist colors
- Glassmorphism specifications (backdrop blur, transparency)
- Typography system optimized for health data
- Animation principles (calming, informative, professional)
- Shadow, spacing, and responsive systems

2. Component Specifications ([health-component-specs.md](#))

- 30+ **health-focused components** including:
 - Layout components (MainLayout, ThreadSidebar, ResizablePanel)
 - Health chat components (MessageBubble, QueryInput, ThinkingIndicator)
 - Medical team visualization (MedicalTeamView, SpecialistCard, TeamConnections)
 - Health results components (HealthMetricCard, VisualizationRenderer)
 - Utility components (ErrorBoundary, LoadingStates, ToastNotifications)
- Detailed specs for each component with states, interactions, and medical context

3. Interactive Prototypes (3 HTML files)

- **Welcome Prototype:** Landing page with animated medical team preview
- **Main App Prototype:** Full interface showing analysis in progress
- **Synthesis Prototype:** Completed analysis with interactive visualizations
- All prototypes feature glassmorphism, smooth animations, and production-quality polish

4. Layout Guidelines ([health-layout-guidelines.md](#))

- Three-panel responsive layout specifications
- Panel dimensions and behaviors
- Responsive breakpoints (mobile/tablet/desktop)
- Keyboard shortcuts and touch gestures
- Performance optimization strategies

5. Animation Specifications ([health-animation-specs.md](#))

- Medical-appropriate motion design
- Specialist activation animations

Chat controls

Claude Opus 4

Powerful, large model for complex challenges [Learn more](#)**Artifacts**

	Health Insight Design System Click to open document • 1 version
	Health Component Specifications Click to open document • 1 version
	Health Welcome Page Prototype Click to open website • 1 version
	Health Main Application Prototype Click to open website • 1 version
	Health Synthesis & Visualization Prototype Click to open website • 1 version
	Health Layout Guidelines Click to open document • 1 version
	Health Animation Specifications Click to open document • 1 version
	Health Visualization Specifications Click to open document • 1 version
	Health Accessibility Guidelines Click to open document • 1 version

Project content

UX Designer Agent

Created by George Vetticaden

Content

User Stories, User Flow Mockups - 7/1/2025	PRD.pdf 209 lines	user-stories.md 273 lines
PDF	MD	MD

Design Package Complete: 8 comprehensive artifacts including 3 interactive HTML prototypes, 25+ component specifications, design system, and accessibility guidelines — everything needed for Claude Code to implement

But the proof was in the prototype. Opening the ux mockup revealed a professional healthcare interface with the multi-agent pattern embodied as a medical team:

The screenshot displays two main sections of the Health Insight Assistant. On the left, a patient consultation for 'Cholesterol Analysis' is shown, detailing a comprehensive analysis of cholesterol medication and lab results from 2013-2025. It includes a 'Direct Answer to Your Question' section, 'Key Findings', and a 'Your Medication Journey and Cholesterol Response' section. A message at the bottom says 'CMO is synthesizing findings...'. On the right, a 'Medical Team' visualization shows a hierarchy of specialists: Dr. Heart (Cardiology 90%), Dr. Pharma (Pharmacy 10%), Dr. Vitality (Chief Medical Officer 90%), Dr. Analy... (Data Analyst 0%), Dr. Prev... (Preventive Med... 30%), and Dr. Horm... (Endocrinology 90%). The visualization indicates 0 waiting, 0 active, and 5 complete tasks. Below the hierarchy, a progress bar shows 'Agent Progress' at 90% and 'Overall Team Progress' at 100%.

High-Fidelity Design Prototype: UX Designer Agent's professional healthcare interface design — transforming Anthropic's multi-agent pattern into an intuitive medical team visualization with CMO Dr. Vitality orchestrating 8 specialists in a consumer-friendly consultation system.

The interface tells a story: Five specialists completed their analysis while two continue working — parallel execution in action. Building such a sophisticated healthcare system typically takes weeks or months, but the 3 Amigo Agents pattern enabled it to be completed in under 3 hours.

The pattern works because each agent adds layers of context that the next one builds upon. The PM Agent's product specifications became the UX Agent's design requirements, resulting in a comprehensive set of documentation. However, even the most detailed specifications and beautiful designs are merely artifacts until they become functional software.

This is where the third amigo — Claude Code — transforms everything into reality.

Claude Code: From Empty Directories to Living System

With comprehensive product specifications from the PM Agent and a complete design system from the UX Agent, Claude Code has everything needed to build the multi-agent health system. This is where the 3 Amigo Agents pattern completes its cycle.

The workspace organization reflects the journey so far:

```
health-insight-system/
└── requirements/
```

```
po-inputs/          # My original vision
pm-outputs/        # PM's specifications
ux-outputs/        # UX's designs
technical-patterns/ # Proven patterns
backend/           # Empty directory
frontend/          # Empty directory
CLAUDE.md          # The implementation guide
```

The `CLAUDE.md` file serves as an implementation guide, directing Claude Code through a methodical three-phase approach—the same way I'd onboard a new engineer to a complex project:

Phase 1: Deep Analysis — Study all product assets, understand the 25+ components, review the pre-built tools

Phase 2: Implementation Planning — Create a comprehensive plan before writing code

Phase 3: Systematic Execution — Build backend, then frontend, with continuous testing

Claude Code's response demonstrates the value of comprehensive context. Rather than diving straight into coding, it first creates a detailed implementation plan:

Frontend Implementation Plan

1. Three-Panel Layout

- Left: Thread sidebar (320px) with search and categories
- Center: Chat interface with message history
- Right: Medical team visualization / Results (45% width)

2. Component Library (25+ Components)

- **Layout:** MainLayout, Header, ThreadSidebar, ResizablePanel
- **Chat:** ChatInterface, MessageList, MessageBubble, QueryInput, ToolCall, ThinkingIndicator
- **Agents:** AgentTeamView, AgentCard, TeamConnections, ProgressIndicator, StatusBadge
- **Visualizations:** VisualizationHistory, QuerySelector, CodeArtifact, VisualizationCard, ExportButton
- **Common:** ErrorBoundary, LoadingStates, EmptyStates, ConfirmDialog, ToastNotifications

3. Glassmorphism Design System

- Glass panels with backdrop blur
- Consistent shadow and border styling
- Gradient background (blue to purple)
- Professional color palette matching specialist themes



4. Real-time Features

- EventSource for SSE connection
- Auto-reconnection with exponential backoff
- Progressive agent status updates
- Live terminal output display
- Animated SVG connections between agents

5. Thread Persistence

- LocalStorage with debounced auto-save
- Thread export/import capabilities
- Query-based visualization history
- Search functionality across threads

Backend Implementation Plan

1. Core FastAPI Application

- FastAPI 0.104.1 with SSE support via sse-starlette
- CORS configuration for frontend communication
- Health check and status endpoints
- Proper error handling middleware

2. Multi-Agent Architecture

- **CMO Agent:** Query complexity analysis (SIMPLE/STANDARD/COMPLEX)
- **8 Specialist Agents:** Cardiology, Lab Analysis, Endocrinology, Data Analytics, Vitality, Internal Medicine, Pharmacology, Preventive Medicine

- **Visualization Agent:** Generates React components with Recharts
- Parallel execution with progressive status updates

3. Thread Management

- UUID v4 for unique thread identification
- Thread persistence with message history
- Query-based visualization tracking
- Thread categories (Today, Yesterday, Past Week, etc.)

4. SSE Streaming Implementation

- GET endpoints with proper headers (X-Accel-Buffering: no)
- Event types: connected, agent_activated, agent_completed, visualization_ready, synthesis_complete
- Reconnection support with event IDs
- Progressive updates for real-time UI feedback



5. Integration with Pre-built Tools

- Snowflake connection utilities
- Health data importer

- Natural language query executor
- Tool registry for Anthropic function calling

Methodical Approach: Claude Code created a comprehensive implementation plan with 100+ specific tasks before writing any code — just like a senior engineer would

The plan covers:

- **Backend:** FastAPI with SSE streaming, CMO orchestrator agent, 8 specialist agents, visualization agent
- **Frontend:** Three-panel layout, 25+ React components, real-time medical team visualization, thread persistence
- **Integration:** Pre-built Snowflake tools, natural language health queries, progressive result streaming

45 minutes later, with some debugging, those empty directories contain a complete application:

```
backend/
├── api/
│   ├── __init__.py
│   ├── chat.py          # SSE streaming endpoints
│   └── threads.py       # Thread management
└── services/
    ├── agents/
    │   ├── cmo/
    │   │   ├── cmo_agent.py      # Chief Medical Officer orchestration
    │   │   ├── cmo_prompts.py   # CMO system prompts
    │   │   └── prompts/         # Task-specific prompts
    │   ├── specialist/
    │   │   ├── specialist_agent.py # 8 medical specialists
    │   │   └── prompts/          # Specialty-specific prompts
    │   └── visualization/
    │       ├── visualization_agent.py # Dynamic chart generation
    │       └── prompts/             # Visualization examples
    ├── streaming/
    │   ├── sse_response.py      # Real-time event streaming
    │   └── __init__.py
    └── health_analyst_service.py # Main orchestration service
    └── models.py               # Data models
15+ more files...
```

```
frontend/
├── src/
    ├── components/
    │   ├── medical-team/
    │   │   ├── AgentCard.tsx        # Specialist status cards
    │   │   ├── MedicalTeamOrgChart.tsx # Team visualization
    │   │   ├── QuerySelector.tsx     # Query-based filtering
    │   │   └── StreamingAnalysis.tsx # Real-time updates
    │   ├── visualizations/
    │   │   ├── DataTable.tsx        # Health metrics display
    │   │   ├── ImportDashboard.tsx   # Data import UI
    │   │   ├── ChatInterface.tsx     # Main chat interaction
    │   │   ├── MessageBubble.tsx      # Message display
    │   │   ├── ThreadSidebar.tsx      # Conversation history
    │   │   ├── ToolCallVisualizer.tsx # Tool execution display
    │   │   └── VisualizationPanel.tsx # Dynamic charts
    │   └── 15+ more components...
    ├── services/
    │   ├── api.ts                # API client
    │   └── sseEventParser.ts      # SSE event handling
    ├── types/
    │   ├── index.ts              # Type definitions
    │   └── medical-team.ts       # Medical team types
    ├── App.tsx                 # Main application
    └── main.tsx                # Entry point
20+ more files...
```

Complete Implementation: Empty directories transform into a full multi-agent health system with sophisticated backend orchestration and comprehensive React frontend

After a few quick iterations with Claude to fix import paths and configuration issues (about 5 minutes), I have both servers running:

```
Quick Start with Make:  
# Install all dependencies  
make install  
  
# Run both backend and frontend  
make dev  
  
Manual Start:  
# Backend  
cd backend  
python -m venv venv  
source venv/bin/activate # Windows: venv\Scripts\activate  
pip install -r requirements.txt  
python main.py  
  
# Frontend (new terminal)  
cd frontend  
npm install  
npm run dev  
  
Docker Deployment:  
docker-compose up
```

⌚ System Architecture

- **Backend:** <http://localhost:8000>
- **Frontend:** <http://localhost:5173>
- **API Docs:** <http://localhost:8000/docs>

The implementation follows all specifications from the requirements documents, including:

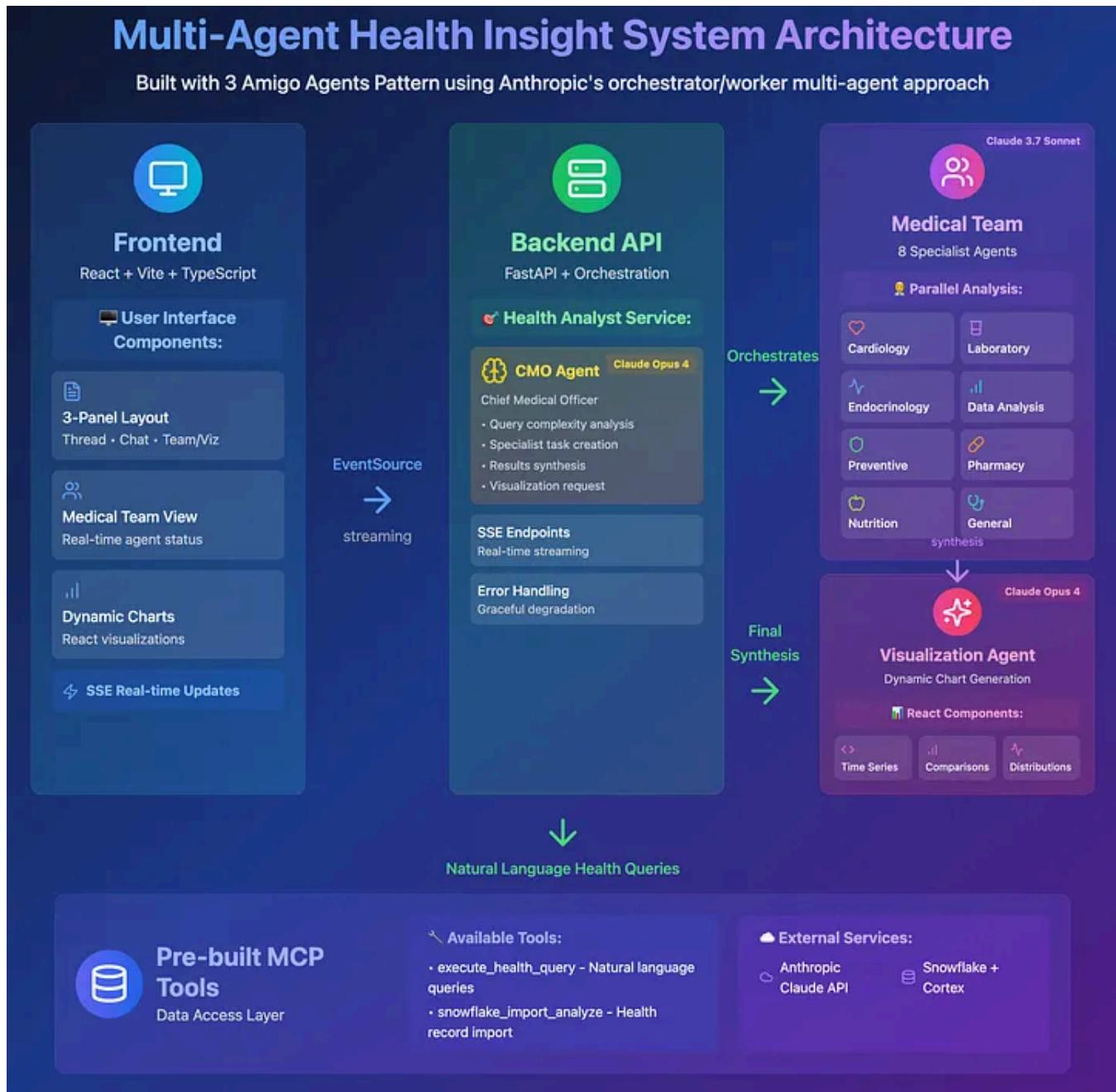
- Exact technology versions (FastAPI 0.104.1, React 18.2.0, Tailwind 3.3.0)
- Multi-agent patterns with CMQ orchestration
- Complete UI matching the provided prototypes
- All production features including error handling and persistence

System Live: Both FastAPI backend and React frontend running successfully after quick debugging — the multi-agent health system is ready for use

The complete system is now live — backend API with multi-agent orchestration, frontend React application, and even interactive API documentation at <http://localhost:8000/docs>. All from empty directories in just 3 hours.

The Architecture Revealed: What the 3 Amigos Actually Built

But what exactly did the 3 Amigo Agents build? Let me show you the architecture of this multi-agent health system:



Enterprise-Grade Architecture in 3 Hours: The sophisticated multi-agent system created by the 3 Amigo Agents — PM Agent designed the orchestration pattern, UX Agent specified the medical interface, and Claude Code implemented real-time streaming with parallel specialist execution.

This architecture diagram reveals the sophisticated orchestration at work. The 3 Amigo Agents didn't just create a simple app — they built an enterprise-grade system implementing Anthropic's multi-agent pattern with:

- **Frontend Layer:** A complete React application with real-time SSE streaming, medical team visualization, and dynamic chart rendering

- **Orchestration Layer:** CMO Agent (Chief Medical Officer) analyzing query complexity and delegating to the right specialists
- **Specialist Layer:** 8 medical domain experts working in parallel — from Dr. Heart analyzing cardiovascular data to Dr. Analytics finding statistical patterns
- **Visualization Layer:** Dynamic React component generation creating health-specific charts on demand
- **Foundation Layer:** Pre-built tools abstracting complex health data queries through natural language

Each architectural decision can be traced back to the comprehensive context provided by the 3 Amigos. The PM Agent specified the orchestrator-worker pattern. The UX Agent designed the medical team visualization. And Claude Code implemented it all with proper SSE streaming, parallel execution, and error handling.

This outcome reflects the real power of the 3 Amigo Agents pattern. Yes, there were a few import issues and configuration tweaks needed — but they took just 5 minutes to resolve with Claude's help. The pattern works because each agent contributed exactly what was needed:

- **The PM Agent** provided clear specifications of what to build — including the multi-agent orchestration architecture you see above
- **The UX Agent** defined how it should look and behave — transforming the technical architecture into a medical team metaphor
- **Claude Code** had all the context to implement it correctly — every layer, every connection, every real-time update

The pattern demonstrates something profound: with the right context and organization, AI can handle complex software development tasks that traditionally require teams and extended timelines. Not through perfection on the first try, but through rapid iteration with comprehensive understanding.

The Revolution Ahead: AI Teams, Not AI Tools

Looking at the architecture above, you're not seeing the output of a single AI assistant — you're seeing the collaborative work of an AI development team. The 3 Amigo Agents pattern represents a fundamental shift in how we build with AI. Instead of humans

writing code with AI assistance, we're orchestrating AI teams that handle entire development lifecycles.

This isn't just faster — it fundamentally changes what's possible for small teams. By orchestrating specialized AI agents, we can now access the collective expertise of an entire product development team. The multi-agent health system proves this: sophisticated backend orchestration, professional frontend design, real-time streaming, parallel processing — all generated in hours, not months.

Each implementation validates the same principle: comprehensive context enables Claude Code to build complex applications with minimal iterations. The architecture you see above — with its orchestrated specialists achieving 90.2% performance gains — would typically require a team of senior engineers weeks to design and implement.

As AI agents become more sophisticated, the pattern will evolve. Imagine extending the architecture with:

- **QA Agents** that create comprehensive test suites for each specialist
- **DevOps Agents** that handle deployment and monitoring of the distributed system
- **Security Agents** that audit the health data handling for HIPAA compliance
- **Performance Agents** that optimize the parallel execution paths

The 3 Amigos will become 5, then 7, then entire AI organizations. But the core insight remains: orchestrated specialization outperforms monolithic approaches — just as the CMO orchestrating medical specialists outperforms a single health analyst.

Conclusion: From Discovery to Revolution

What started as an effort to implement Anthropic's multi-agent architecture for my health application became something much bigger. The 3 Amigo Agents pattern didn't just solve my immediate problem — it transformed how I approach every development project.

The architecture diagram above isn't just a technical achievement — it's a glimpse into the future of software development. A future where a small team of developers orchestrates AI teams to build systems that previously required entire engineering departments.

This is the future Anthropic's research points toward: not bigger models doing everything, but orchestrated specialists achieving the impossible. The 3 Amigo Agents pattern is just the beginning of this revolution.

In Part 2, I'll delve into the multi-agent health system itself — how each specialist in the architecture functions, what the orchestration looks like in action, and the comprehensive evaluation framework that demonstrates these approaches deliver on their promises.

But you don't need to wait. You have the pattern now. You've seen the architecture it can produce. Your next AI project — as complex as the multi-agent system above — is just ~3 hours away.