



Los Angeles

Building With Agentic AI

Emily Anderson

GDG Culver City Organizer/WTM Ambassador



Google
Developer
Groups



2025





AGENDA

A few key points

What it is

Why it's important

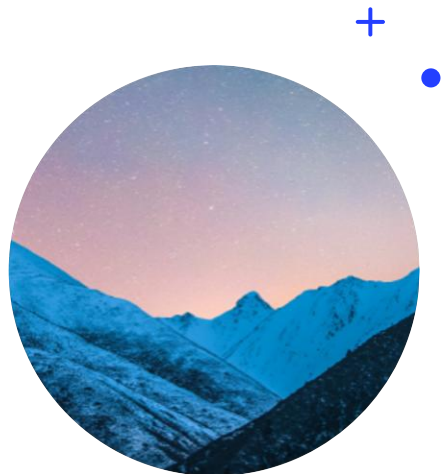
Examples

Demos



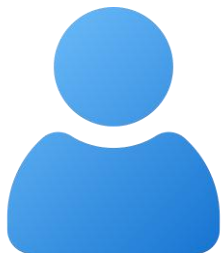
A FEW KEY POINTS (WIP)

- **Empowerment over replacement** – AI is here to supercharge our skills, not take our jobs.
- **Mindset matters** – Continuous learning is critical to leveraging AI effectively.
- **Powerful but not magic** – Amazing capabilities, but there's a learning curve.
- **Context is everything** – AI output quality depends on what you feed it.



WHAT IT IS

User



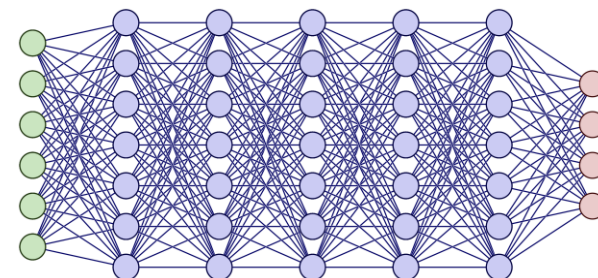
- Request
- Question
- Task
- Etc.

**Agent/Orchestration Layer
+ IDE/Codebase**



- Task creation and orchestration
- Context management
- Architect
- Code
- Test
- Debug
- Document
- Etc.

Generative AI Model





WHY IT'S IMPORTANT

- Capabilities and quality of output increasing rapidly (models and tooling)
- Major tech companies already generating substantial % of code with AI (add "as of NOW" stats)
- Dramatically increases productivity (with caveats)
- Reduces reliance on vendors
- Potential for more build vs. buy
- Build, market, sell our own products / IP
- Something about not falling behind (we need to be able to attract the right talent)



EXAMPLES

Build a web application called "PRODUCTIZER" with the following specifications:

****Core Functionality:****

- Create an AI-powered product analysis tool that takes product context as input and generates comprehensive product summaries
- The output should include: features, key technologies, high-level architectures, multi-tenancy factors, and other relevant product details
- Use Anthropic Claude Sonnet 4 as the LLM for processing and analysis

****Technical Stack:****

- Backend: Node.js with Express framework
- Frontend: Vanilla JavaScript with TypeScript
- No HTTPS required (demo/development environment)
- Simple text input interface initially (single textarea for product context input)

****UI/UX Design:****

- Theme: Monster or dinosaur "chomping through" product context
- Visual metaphor: The app should feel like a creature consuming and digesting product information to produce insights
- Consider monster/dinosaur imagery, animations, or visual elements that reinforce this theme

****Development Approach:****

- Start with basic functionality: text input → AI processing → formatted output display
- Structure the codebase to easily accommodate future enhancements
- Plan for future features like document upload (Word docs, PDFs, etc.) but don't implement yet

****Project Setup:****

- Initialize as a new Node.js project with proper TypeScript configuration
- Set up Express server with appropriate routes for the AI processing
- Create a clean, responsive frontend interface
- Implement proper error handling and loading states

****Deliverables:****

1. Working web application with text input and AI-generated product summaries
2. Monster/dinosaur themed UI that's both functional and engaging
3. Clean, maintainable code structure ready for future enhancements





EXAMPLES

I'll help you build the PRODUCTIZER web application with the monster/dinosaur theme and AI-powered product analysis. Let me start by gathering information about the current workspace and then create a detailed plan.

• Read directory

Great! I'm starting with a clean workspace. Let me create a detailed plan for building the PRODUCTIZER application.

Detailed Plan for PRODUCTIZER Web Application

Project Structure:

```
productizer/  
├─ package.json  
├─ tsconfig.json  
├─ server/  
│ └─ src/  
│   │ └─ app.ts (Express server setup)  
│   │ └─ routes/  
│   │   └─ analyze.ts (AI processing route)  
│   └─ services/  
│     └─ claude.ts (Anthropic Claude integration)  
│     └─ types/  
│       └─ index.ts (TypeScript interfaces)  
└─ dist/ (compiled JS output)  
├─ client/  
│ └─ src/  
│   │ └─ index.html  
│   │ └─ styles/  
│   │   └─ main.css (monster/dinosaur themed styles)  
│   └─ scripts/  
│     └─ main.ts (main application logic)  
│     └─ api.ts (API communication)  
│     └─ assets/  
│       └─ images/ (monster/dinosaur graphics)  
└─ dist/ (built frontend assets)  
└─ README.md
```




EXAMPLES

Implementation Steps:

1. Initialize Node.js project with TypeScript

- Set up package.json with necessary dependencies
- Configure TypeScript for both server and client
- Set up build scripts

2. Backend Development

- Create Express server with CORS support
- Implement `/api/analyze` endpoint for product analysis
- Integrate Anthropic Claude Sonnet 4 API
- Add proper error handling and validation

3. Frontend Development

- Create monster/dinosaur themed HTML structure
- Implement CSS with creature "chomping" animations
- Build TypeScript modules for API communication
- Add loading states and error handling

4. AI Integration

- Configure Claude API with appropriate prompts for product analysis
- Structure output to include: features, technologies, architecture, multi-tenancy factors
- Implement proper response formatting

5. Testing and Polish

- Test the complete flow
- Refine the monster theme and animations
- Ensure responsive design



EXAMPLES

Let me start implementing this plan:

- > Created file package.json
- > Created file tsconfig.json
- > Created file tsconfig.json server
- > Created file tsconfig.json client
- > Created file index.ts server/src/types
- > Created file claude.ts server/src/services
- > Created file analyze.ts server/src/routes
- > Created file app.ts server/src

Now let me create the frontend files with the monster/dinosaur theme:

- > Created file index.html client/src
- > Created file main.css client/src/styles
- > Created file api.ts client/src/scripts
- > Created file main.ts client/src/scripts

Now let me create the environment file template and README:



EXAMPLES

Terminal

```
$ Invoke-WebRequest -Uri "http://localhost:3001/api/analyze" -Method POST -Headers @{"Content-Type"="application/json"} -Body '{"productContext":"A web application for managing tasks and projects with user authentication"}'
```

Perfect! The API is working correctly and properly handling the missing API key with a user-friendly error message.

🚀 PRODUCTIZER Application Successfully Built!

I've successfully created the PRODUCTIZER web application with all the requested features. Here's what has been delivered:

✅ Core Functionality Implemented:

- AI-powered product analysis tool that processes product context input
- Comprehensive output generation including features, technologies, architecture, and multi-tenancy factors
- Anthropic Claude Sonnet 4 integration ready to use (just needs API key)
- Robust error handling and validation

✅ Technical Stack:

- Backend: Node.js with Express framework ✓
- Frontend: Vanilla JavaScript with TypeScript ✓
- No HTTPS required (development environment) ✓
- Simple text input interface with textarea ✓

✅ Monster/Dinosaur Theme:

- 🦖 Monster branding throughout the application
- Visual metaphor: Monster "chomping through" product context
- Animated elements: Blinking eyes, floating monster, chomping mouth
- Monster-themed colors: Dark background with orange/gold accents
- Playful typography: Creepster font for headers, Fredoka for body text

✅ Project Structure:

- Clean, maintainable codebase with TypeScript
- Modular architecture ready for future enhancements
- Proper error handling and loading states
- Responsive design that works on desktop and mobile

🔧 How to Use:

1. Set up your Anthropic API key:

```
# Edit .env file and replace with your actual API key
ANTHROPIC_API_KEY=your_actual_api_key_here
```

Copy ...

2. The application is already running at: <http://localhost:3001>

3. Test the monster in action:



EXAMPLES



The Monster That Devours Product Context!



 **Feed the Monster**

Drop your product context into the monster's mouth and watch it get chomped into comprehensive insights!

Describe your product, features, technology stack, architecture, or any product-related information here... The monster is hungry for details! 🦖

0 / 800,000 characters

 **CHOMP & ANALYZE**



DEMOS

Etc.



+



Q&A