Kahuna 🗿 Labs

Stop vibing with your code. **Start vibing with requirements.**

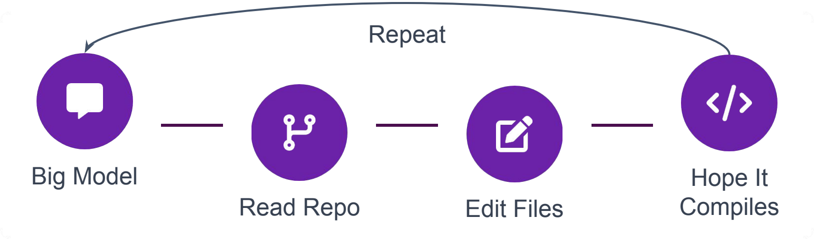
Let us develop your application

We've built the engine that turns specs into full systems — and now we're looking for the partner who will turn that into a movement.

Why "Vibe Coding" Doesn't Scale

The Fragile Loop

Most AI dev platforms operate on the same fragile loop:



Structural Constraints

- A non-trivial codebase is **orders of magnitude larger** than a single LLM context
- Tools keep re-reading and re-editing **partial views** of the system
- Model cannot truly "hold" the whole architecture in its head
  * They might end up breaking working features and slide into architectural drift.
  * Humans must constantly step in and babysit the agent.
  * Reviews describe projects abandoned after burning through credits without reaching a stable app.

Consequences

⛔ **The results are hard to be used in a corporate context**

⛔ **Wasted Time & Budget**
Substantial waste for serious projects, hard to integrate into corporate delivery pipelines

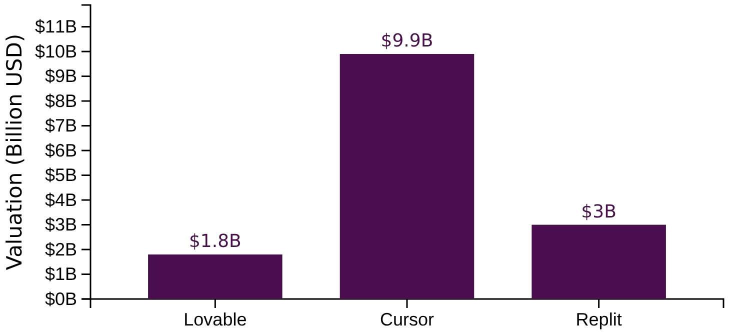
⛔ **Architectural Drift**
Risk of breaking working features when re-editing partial views

⛔ **Tech Stack Lock-in**
Platforms lock into controlled tech stacks (e.g., Lovable → React/Vite/Tailwind/Supabase)

💡 **Speaker note:** The problem isn't that these companies are incompetent; it's that they share the same bet: rely exclusively on the model and iterate in loops.

The Market Is Hot – and Still Open

AI Coding Tools Market



✓ Clear Proof

AI coding tools are a multi-billion market with serious ARR and valuations:

Devs and organizations **pay real money** for AI-assisted development.

The Opening

No Tool Exists that can:

- Turn large, enterprise-level requirements into coherent systems
- Reliably control flow and cost on complex builds

"Late in the Game" Opening

Expectations are set, budgets exist — but architecture and business models are still unsettled.

Speaker note: The market has already been educated for us. Our job is not to convince people AI dev tools matter. Our job is to show that a different *architecture* is needed for serious systems.

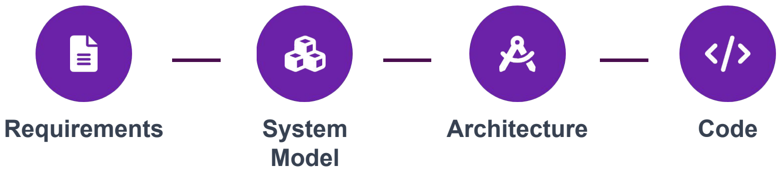
Our Thesis: Build from Requirements, Not from the Repo

Core Belief

Product and leadership teams think in requirements and outcomes – not "edit `main.js`."

> **The primitive of software delivery should be the requirements document, not the code.**

Spec-to-System Pipeline



Requirements

System Model

Architecture

Code

How It Works

Parallel Processing : Our algorithm breaks down the problem in a way that achieves effective parallelization at every stage

Restricted Scope: Each stage is scoped and executed only within the relevant global context.

Defined Pipeline : Each stage runs in a controlled pipeline with of pre-ordered queries with bounded cost and predictable outputs

Key Advantages

Coherent Architecture : Reads structured specs and builds a coherent architecture first

Reduced Cognitive Load : Working on specs rather than on code and always within the correct context sensibly lesser the mnemonic burden on the LLM

Multi-tier Generation : Generates complete systems with backend, frontend, database, and API layers

More predictable results and and measurable cost control **much less architectural drift**

LLM Limits Are Real: how do we approach them.

Limit 1: LLM Generated code is not perfect

- We don't think that costly **self-correction loops** over limited codebases are real **added value** for developers.
- They would rather fix a **fully scaffolded integrated codebase**, produced by several LLMs orchestrated in parallel, than pay for a black-box autopilot that spends most of their money trying to correct itself.

In Other Words, developers would prefer a jet engine they can steer themselves, rather than an expensive black-box autopilot

Empirical reality:

- Most devs use AI, but **don't fully trust it** already.
- Copilot-style suggestions are welcomed even though they're often wrong.
- Tools like **v0.dev by Vercel** are explicitly **"generate then tweak"** and are accepted on those terms.

Limit 2: LLMs have limited understanding over certain technologies

Officially supported stacks and techs first

- We start with a **short list of "blessed" stacks and technologies**
  e.g. React + Node + Postgres; Solidity + Hardhat + Node backend;
  allowing for recombinations of reliable tech
  React Native; Unity; Firebase, etc.
  making the list of supported stacks **far less opinionated and more flexible** than fixed-stack tools.
- This would let us inject **stack-specific prompts/templates** and, later, specialized models into the pipeline.

Advanced mode for power users

An advanced mode with:

- capability self-check
- optional strategic RAG ingestion over the user's own knowledge base
- potentially usage of custom trained LLMs

In this mode, users know they must give **more precise technical requirements** and accept more debugging — in exchange for deeper integration with their preferred stack.

The Engine (Built) – Not Just a Demo

Requirements Studio

Guided interface to turn fuzzy ideas into structured specs.

This is not core to the technology: Is in Early-stage prototype exists and can be extended; this is **not** a major technical risk.

System Generation Engine

We deliberately went **engine first, product second**
Today, the engine can:

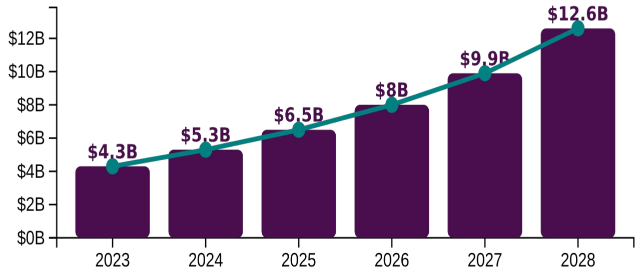
- Can interpret requirements documents of up to ≈2.5k lines
- Generates up to ≈17k lines of code in a single coherent run

A public prototype exists as a proof point; the core engine has already been tested on non-trivial specs.

https://www.kahuna-labs.de

A Rare Timing Window

Market Growth



24% CAGR growth in AI dev tools market

> Billions are already being spent on tools that **assist** coding but don't **re-architect delivery**.

Enterprise Adoption

Over the last 2 years enterprises moved from "Is this real?" to "Which vendor do we standardize on?"

Market Gap

No tool exists for requirements-to-system automation, leaving a critical gap in the market

The Window Is:

- Late but not closed
- Before architecture ossifies
- First-mover advantage in spec-to-system layer

What's Missing

Market Architecture

Need partners to define customer strategy

The Right GTM Partner Can

Turn Technology into Category

Define "spec-to-system" as the next layer after Copilot

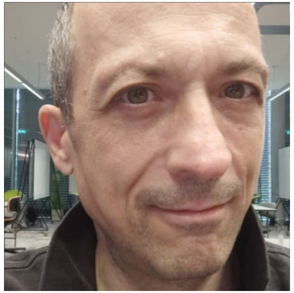
Build Design Partnerships

Create first 3–5 strategic partnerships

Own Narrative & Channels

Drive GTM while we refine the engine

Team – Deep Technical Founder in Place



Alberto Marabini

Technical / Product Founder

25+ Years Experience
Enterprise Systems Design & Delivery

LLM-Powered Systems
Multi-stage agentic pipelines & RAG

Professional Background

Alberto has designed and delivered enterprise systems across banking, pharma, manufacturing, HR, media, and telecom industries.

Full-Stack Architecture

Multi-tier backends, complex HTML5/JS UIs, React/Angular/Salesforce, cloud & DevOps

Enterprise Delivery

Leading technical architect and engineer at Cisco, Wells Fargo, Genentech/Roche, Novartis, TriNet, Sky

Who I'm Building This With

Primary Ask

A business/GTM partner at co-founder level who can:

Go-to-Market

- ✓ ICP, positioning, pricing

Customer Dev

- ✓ Startups, agencies, enterprises

Fundraising

- ✓ Shape rounds, manage relationships

Commercial Ops

- ✓ Sales motions, partnerships, hiring

What I Bring

Deep Technical + Product
Evolving the engine, designing workflow, working with early users

Enterprise Experience
Understanding how large orgs think about systems, risk, and delivery

Proven Execution
Built the core engine and early prototypes solo