SEED / SERIES A · 2026

# Shipwise55

## Automating the Chief Product Officer for AI-built software.

The specification & memory layer that turns product thinking into editable text and visual system graphs — so AI can actually ship production software instead of crashing it. The bottleneck where vibe-coders hallucinated and broke backends is solved.

**shipwise55.com · moving to 11builders.com**

# Vibe-coding has no CPO — so it crashes production.

**TODAY'S TOOLS OPTIMIZE FOR**

- Speed
- Instant demos
- Fast prototyping

**REAL ENGINEERING REQUIRES**

- Memory
- Architecture
- Specifications
- Long-term consistency
- A real CPO function

**WHAT'S ACTUALLY MISSING**

The CPO role — deciding what to build, designing the architecture, owning NFRs and edge cases — is being skipped. Tools like Lovable hallucinate backends and crash in production because there is no persistent spec or system memory to reason against.

# Bigger context windows alone don't solve it.

**THE INDUSTRY RESPONSE**

- Larger context windows
- Brute-force token scaling
- More expensive inference

**WHAT A USEFUL SYSTEM MUST DO**

- Know what matters
- Remember past decisions
- Retrieve exact architecture state
- Localize changes
- Reason hierarchically
- Preserve consistency over time

*The real bottleneck is not token count. It is persistent, structured memory.*

# The missing layer in AI software engineering.

**CURRENT PRODUCTS**

Lovable, Cursor, Replit, Devin, Windsurf — all focus on:

— Generating code quickly

— Compressing developer workflows

— Accelerating implementation

**WHAT NONE SOLVE**

— Automated CPO function

— Persistent architectural memory

— Specification-driven engineering

— Long-context orchestration

— Enterprise-grade maintainability

# Automate the CPO. Make software maintainable in text and graphs.

### SHIPWISE55 AUTOMATES THE CPO

- Requirements extraction from PRDs, transcripts, ideas
- Architecture and service decomposition
- Non-functional requirements and constraints
- Edge cases and dependency mapping
- Spec-first validation before generation

### EDITABLE INTERNALS — TEXT + GRAPH

- Every decision stored as human-readable text
- Every relationship visible as a navigable graph
- Localized edits — change one node, regen one module
- No full-context rewrites, no drift
- Maintainable production software, not demos

# The Shipwise55 workflow.

**STEP 01 — UPLOAD**

Product idea, PRD, meeting notes, architecture docs.

**STEP 02 — EXTRACT**

Use cases, actors, requirements, permissions, APIs, edge cases.

**STEP 03 — GENERATE**

Specifications, architecture maps, system diagrams, roadmap.

**STEP 04 — REVIEW**

User reviews and edits specs. Architecture validated before code.

**STEP 05 — BUILD**

AI agents generate modular implementations against validated spec.

**STEP 06 — PERSIST**

Living memory layer tracks evolution. Edits localize to modules.

# Specifications are cheaper than code.

### WHY TRADITIONAL ENGINEERING WORKS

— Humans review requirements before implementation

— Architecture decisions happen before coding

— Stakeholders understand system behavior

### MODERN AI CODING SKIPS THIS

— Uncontrolled generation

— Hallucinated dependencies

— Fragile systems that crash in production

### SHIPWISE55 RESTORES

— Architectural reasoning

— Human-readable system understanding

— Iterative requirement refinement

— Localized system updates

# The Hierarchical Memory Transformer.

**NOT JUST**

— Pure sparse attention

— Pure RAG

— Pure LongRoPE scaling

— Brute-force context windows

**INSTEAD — FIVE COOPERATING LAYERS**

— Local transformer reasoning

— Learned long-term memory

— Sparse block routing

— Semantic KV-cache compression

— Exact retrieval fallback

*Massive effective context windows — with architectural consistency.*

# How we solve the context window problem.

**01** **Local active reasoning**
High-quality short-range coding reasoning.

**02** **Specification memory**
Persistent understanding of product requirements.

**03** **Architecture graph memory**
Services, APIs, flows, dependencies as a living map.

**04** **Sparse retrieval routing**
Selective retrieval — slashing tokens and hallucinations.

**05** **Exact source retrieval**
Precise retrieval of implementation details on demand.

# The production crash problem — solved at the root.

### WHAT KILLED VIBE-CODING IN PRODUCTION

- Lovable-style hallucinated backends and crashes
- Models forget prior decisions every session
- Codebases drift and become unstable
- Small changes trigger cascading regressions
- Non-technical owners lose all visibility

### WHAT SHIPWISE55 ACTUALLY DELIVERS

- An always-current spec in plain English
- An editable system graph of services, APIs, flows
- Localized regeneration — change one node, ship one module
- Architecture-aware coding agents that don't drift
- Production-grade software you can maintain

*The bottleneck wasn't the model. It was the absence of a CPO and a persistent system memory. Shipwise55 supplies both.*

# Docs as the source of truth. Code as a side-effect.

### TODAY — LOVABLE, REPLIT, VIBE-CODING

- Prompt → code, no internals exposed
- Non-developers cannot see under the hood
- Amendments overload the context window
- Every fix introduces new bugs
- Backends hallucinate and crash past a threshold
- No shared artifact between stakeholders and AI

### TOMORROW — SHIPWISE55

- LLMs build comprehensive system documentation
- Every actor, flow, API, edge case in plain English
- Review and amend a sentence before any code tokens
- Edits are localized to affected modules
- Code runs against a stable architecture
- Founders, stakeholders, AI read the same document

**THE ECONOMIC ARGUMENT**

A spec is dramatically cheaper than code — for humans and LLMs. Reviewing before shipping saves man-hours and tokens. Localized edits make code changes immune to hallucinations and context limits.

# Existing AI coding tools are stateless.

| COMPANY | MAIN FOCUS | LIMITATION / EDGE |
|---|---|---|
| **Lovable** | Rapid generation | Hallucinates backends, crashes in production |
| **Cursor** | Developer productivity | Limited long-term context consistency |
| **Replit** | Fast prototyping | Architecture drift over time |
| **Devin** | Autonomous execution | Weak persistent system reasoning |
| **Windsurf** | Coding acceleration | Prompt-centric workflows |
| **Shipwise55** | Automated CPO + persistent memory | Production-grade, maintainable, scalable |

# Deep infrastructure moat.

**SHIPWISE55 IP**

- Automated CPO / spec extraction engine
- Proprietary hierarchical memory architecture
- Persistent architecture graph
- Long-context orchestration engine
- Modular AI execution framework
- Enterprise architecture consistency engine

**COMPETITORS OPTIMIZE FOR**

- UI
- Generation speed
- Wrapper experiences

**WE OPTIMIZE FOR**

- Memory systems
- Architectural reasoning
- Persistent state
- Scalable software evolution

# Timing is perfect.

**1**

LLMs are now capable enough for meaningful software generation.

**2**

Context limitations are the dominant bottleneck.

**3**

Enterprises need maintainability and governance.

# A massive emerging market.

**MARKETS IMPACTED**

- AI coding tools
- Enterprise software engineering
- AI developer infrastructure
- Product management platforms
- Enterprise architecture systems
- AI agent infrastructure

**EVERY LARGE ENTERPRISE WILL NEED**

- Persistent AI memory
- Architecture-aware reasoning
- Modular AI orchestration
- An automated CPO function

# Multi-layer revenue model.

### SaaS subscriptions

Founders, startups, product teams.

### Enterprise platform licenses

Large engineering orgs.

### Usage-based orchestration

AI generation and memory compute.

### API infrastructure

Persistent memory APIs for AI agents.

### Private enterprise deployment

Security-sensitive orgs.

### Architecture memory cloud

Long-term persistent system state.

# Phased market expansion.

### Phase 1 — Founders & indie builders

AI apps become unstable fast. We provide structured specs and stable modular generation.

### Phase 2 — Product teams & CTOs

Scaling AI-generated systems. We provide persistent architecture and maintainability.

### Phase 3 — Enterprise engineering

Governance and consistency. We provide long-context orchestration and system memory.

### Phase 4 — AI-native enterprises

Shipwise55 becomes infrastructure for autonomous software evolution.

# Key technical metrics.

### WE AIM TO DEMONSTRATE

- Reduced hallucinations
- Lower regression rates
- Reduced token consumption
- Better architecture consistency
- Better long-session memory retention
- Improved large-repo task completion

### BENCHMARK CATEGORIES

- Large repository modification
- Architecture drift testing
- Multi-session consistency
- Long-running agent memory
- Dependency resolution accuracy
- Enterprise workflow orchestration

# The path to AI-native infrastructure.

| | |
|---|---|
| 2026 | Specification-first AI engineering platform. |
| 2027 | Persistent memory architecture engine. |
| 2028 | Long-context autonomous coding agents. |
| 2029 | Enterprise AI-native software infrastructure. |
| FUTURE | Self-maintaining enterprise software systems. |

# Anton Kravchenko, Founder & CEO.

BACKGROUND

— Fintech and Web3 infrastructure expertise
— Complex trading and financial systems
— Deep systems thinking
— Long-term vision around AI infrastructure

REACH ANTON DIRECTLY

**WhatsApp: +351 932 968 173**

wa.me/351932968173

*AI can generate code quickly. But it cannot persistently reason about evolving systems. Shipwise55 was created to automate the CPO role and supply the memory architecture missing from AI-generated software.*

# The future of software is automated CPO + persistent memory.

The current generation of AI tools helps humans write code. The next generation will:

- Decide what to build (the CPO function)
- Remember systems persistently
- Evolve architecture in text and graphs
- Maintain production software autonomously

# Software engineering needs a CPO, memory, and structure.

Current AI tools generate code. Shipwise55 enables AI to:

- Act as the Chief Product Officer
- Reason architecturally
- Remember persistently
- Evolve systems safely in text and graphs
- Ship production-grade software reliably

## The future of software engineering is
## an automated CPO with persistent system memory.

shipwise55.com · WhatsApp Anton: +351 932 968 173 · moving to 11builders.com