THE ORCHESTRATOR · QUALITY BRIEF

Why Our Software Holds Up

What you get when software is built by a full AI engineering team — not a single developer with an AI assistant working alone.

12	8	4	100%
SPECIALIST ROLES	WORKFLOW STAGES	GATES THAT REJECT WORK	REQUIREMENTS VERIFIED

Most AI-assisted software is built the same way — one general-purpose assistant writing code in a single pass, checked by one busy developer. It is fast, but **speed hides a real risk**. AI is fluent and confident: it can build the wrong thing convincingly, quietly skip the unglamorous work, and — because the same assistant writes both the code and the tests that check it — produce tests that pass even when the feature is broken.

The **Orchestrator** is built differently. Every request runs through a team of twelve AI specialists and a series of independent quality gates — the same disciplined process a strong engineering organization uses — before anything reaches you.

A TYPICAL APPROACH VS. THE ORCHESTRATOR

	A DEVELOPER WITH AN AI ASSISTANT	THE ORCHESTRATOR
Who builds it	✗ One generalist AI filling every role at once	✓ **Twelve specialists** — product, requirements, architecture, design, three development disciplines, and quality
Before any code is written	✗ Often nothing — the AI jumps straight to writing code	✓ The **business goal, detailed requirements, architecture, and user experience** are all defined first
Reviews	✗ One developer, one look	✓ **Multiple independent reviewers** — any one can reject the work and send it back to be fixed
Proof the work is correct	✗ “It looks finished”	✓ Every requirement is a **tracked item with a pass/fail verdict and evidence** — a single failure blocks release
Testing	✗ The same AI writes the code and its own tests together, with nothing checking it	✓ A **separate quality team** treats the code as AI-written and hunts for tests that would pass even if the feature were broken
When it counts as “done”	✗ When it runs and looks right on the surface	✓ **Production-ready** — no placeholders, no TODOs, no unfinished edges; security or accessibility defects are automatic rejections
Consistency	✗ Varies with the developer, the day, and the deadline	✓ **Identical rigor** on every task, every time
What you receive	✗ Code	✓ A **reviewed, tested, documented change** with a plain-language summary of what changed and why

BUILT IN AT EVERY STEP — NOT BOLTED ON AFTERWARD

Security hardening | Accessibility standards | Up-to-date technical knowledge | Complete documentation | Requirement traceability

HOW A REQUEST BECOMES FINISHED SOFTWARE

01 Define › 02 Specify › 03 Design › 04 Plan the experience › 05 Build › 06 Review › 07 Test & verify › 08 Document & package

*At four points along the way, a reviewer can **halt the work and send it back**. Nothing advances until it is right.*

WHAT THIS MEANS FOR YOU

You get what you actually asked for

Requirements are written down and verified one by one — not guessed at and quietly assumed.

Problems surface before delivery

Independent review at every stage means defects are caught inside our process — not in your hands.

The standard never slips

Every project gets the full, identical treatment — whether it is the first or the hundredth.

<