



DAEDALUS AI


The Vision — Daedalus AI: A New Era of Autonomous Software Development

Welcome to Daedalus AI, my vision for transforming software creation through true autonomy and intelligence. This platform is not just another workflow tool. Daedalus AI is designed to be a generative development environment, fundamentally differing from platforms like n8n or Zapier. Instead of simply coordinating or wiring together APIs and services, Daedalus AI is capable of building the services themselves, end-to-end—from a single prompt. Powered by the Trae AI IDE, Daedalus leverages agentic workflows to turn ideas into full-stack, production-ready web applications, documented and deployable out of the box.

Setting Daedalus Apart from the Rest

What really makes Daedalus AI revolutionary is its ability to create, not merely connect. Using advanced multi-agent architecture, Daedalus can take a high-level requirement and architect, implement, and integrate all the moving parts of a modern web application. The system produces everything: backend APIs, databases, frontend code, config files, Docker infrastructure, and AI-generated README documentation. It represents a leap beyond automation—delivering real, autonomous software engineering as a service.





Journey to Our Innovative "Compiler & Executor" Model



Extensive Experimentation

After extensive experimentation, iterations, and lessons learned, we developed a robust "Compiler and Executor" model powered by two specialized agents.



Maximized Reliability

This model maximizes reliability and minimizes error-prone complexity.



Autonomous Workflow

It mirrored the real-world software development process but was distilled into a digital, fully autonomous workflow.



Meet The @Architect Agent — The Master Planner



Analyzes Requirements

When given a project request, it acts like a virtual chief architect—analyzing the requirements, making architectural and stack decisions.



Compiles Project Rules

Ultimately compiling a comprehensive, executable [project_rules.md](#) file.

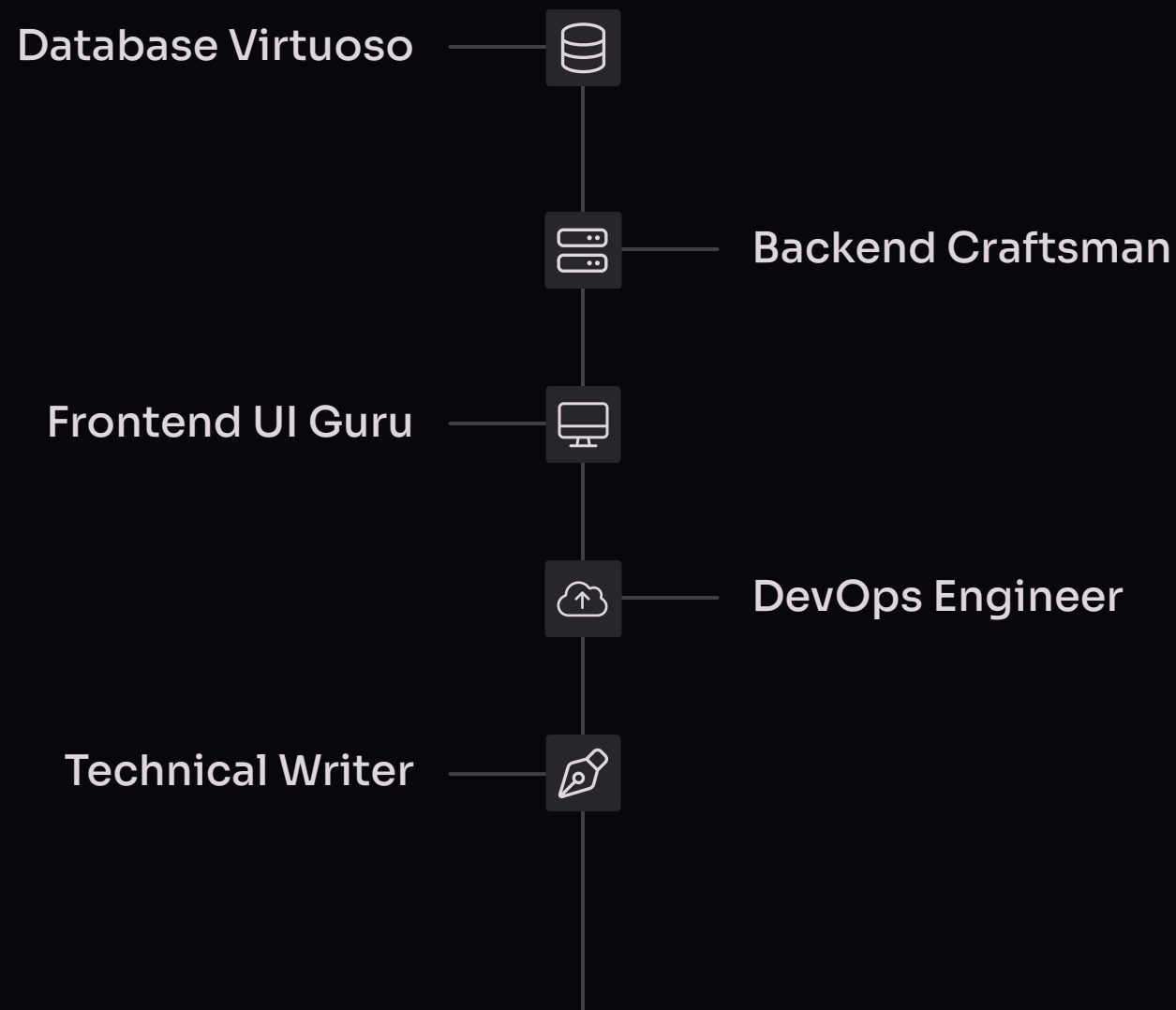


Embeds Expert Personas

This file doesn't just provide steps; it embeds expert personas for every phase—defining standards for database design, backend structure, UI/UX, DevOps, and even documentation.

The @Architect ensures every stage of software creation follows best practices and industry standards, creating a blueprint that leaves nothing to chance.

Meet The @Code_Generator Agent — The Relentless Builder



Once the build plan is set, the **@Code_Generator** takes over as the self-orchestrating workhorse. Unlike a traditional agent that stops or summarizes after each step, this agent works through each phase continuously, without pause. It transforms into a Database Virtuoso, Backend Craftsman, Frontend UI Guru, DevOps Engineer, and finally, a Technical Writer—all thanks to precise persona instructions embedded in the plan. It generates all project artifacts and documentation, ensuring the result is coherent, high-quality, and fully operational.

The Grief of an Unrealized Vision: Striving for a Digital Engineering Team

Despite these successes, I carry the grief of not realizing my most ambitious vision. I aimed to create a digital "software team" of distinct, parallel agents—one for the frontend, backend, database, and DevOps—mirroring real project teams, capable of scaling up and running complex builds in a truly modular and simultaneous fashion. I tried hundreds of syntaxes and invocation models in [project_rules.md](#) to enable this, but the system always fell back on narration—agents would describe intended actions instead of actually calling one another. This technical barrier prevented real agent-to-agent orchestration.

Frontend Agent

Responsible for user interface development.

Backend Agent

Handles server-side logic and APIs.

Database Agent

Manages data storage and retrieval.

DevOps Agent

Oversees deployment and infrastructure.

Technical Hurdles and My Creative Pivot

This limitation informed my creative pivot. Instead of abandoning the project, I adapted by consolidating the orchestration within a single, multi-talented agent that could step into any role as needed, ensuring continuity and success. This self-orchestrating approach, while less modular than my original vision, brought reliability and clarity. Yet, this journey through failed invocations and persistent errors only deepened my understanding of agentic design and my desire to push these boundaries further.

Identify Limitation



Creative Pivot



Ensure Continuity



Consolidate Orchestration



The Road Ahead — My Excitement for Trae IDE and the Future

Despite the challenges, I'm more passionate than ever about working with Trae AI IDE to enable **real-time, true agentic workflows**. My future plans are bold: I want to re-enable multi-agent parallelism, possibly by orchestrating terminal calls externally and integrating Helper Agents to handle framework setup or The Editing Agent for automated fixes to detected errors. I'm excited to collaborate, innovate, and transform autonomous software engineering—turning today's hurdles into tomorrow's breakthroughs. The journey isn't over; it's just begun, and Daedalus AI is ready to lead the way.

Re-enable Multi-Agent Parallelism

Orchestrating terminal calls externally.

Integrate Helper Agents

To handle framework setup.

Implement Editing Agent

For automated fixes to detected errors.