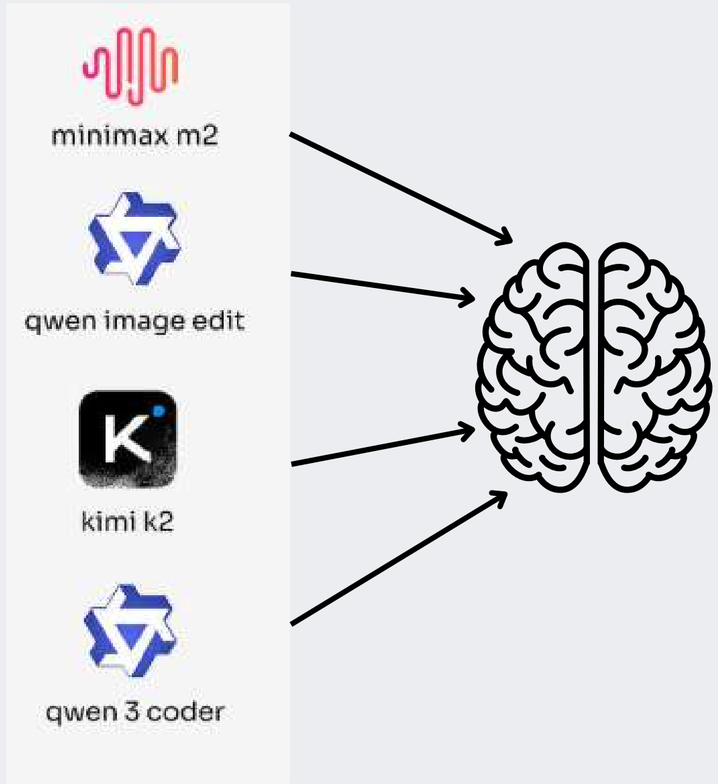
OPEN SOURCE • LOCAL FIRST

LocalForge

The Intelligent Self-Hosted AI Control Plane

- One line changed. Multiple local models. Intelligent routing.

100%	No
Data Privacy	Cloud Latency



THE PROBLEM

Open source AI is exploding but using multiple models is a nightmare

🔍 No Smart Discovery

Manually hunting GGUF files on HuggingFace and guessing if they fit your VRAM

⚡ No Routing

Sending every query to one model — wasting GPU on simple tasks or failing on hard ones

🧠 No Memory

Static benchmarks don't reflect YOUR workload — no learning from real usage

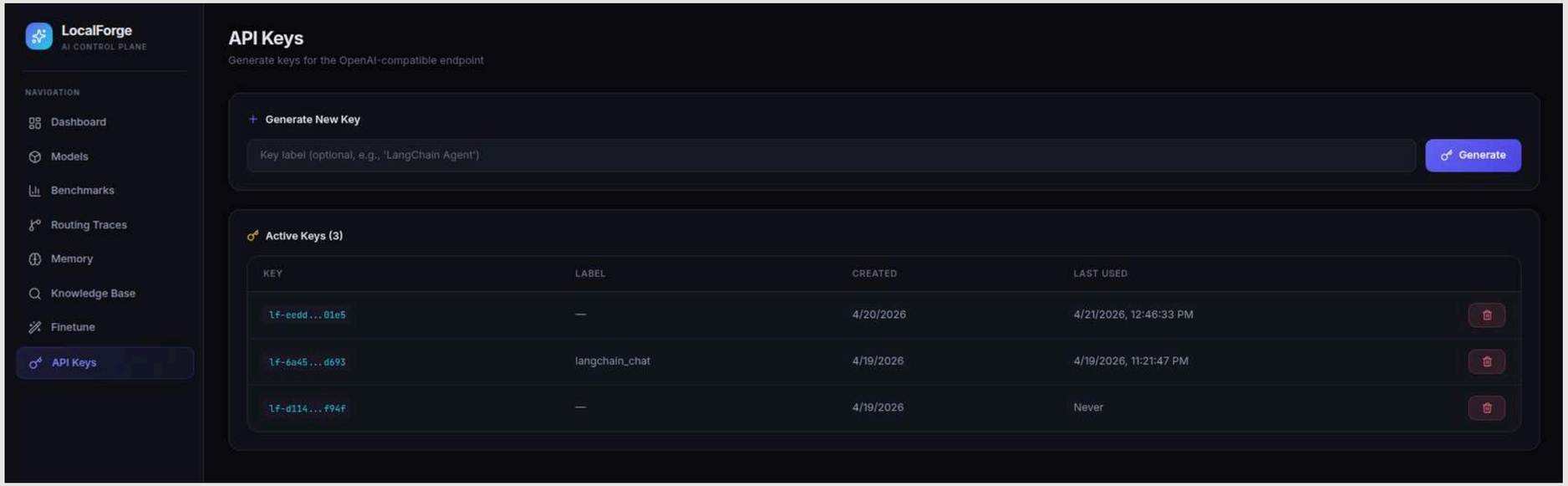
🔧 Broken Integrations

Rewriting LangChain, AutoGen, CrewAI integrations every time you switch models

Result: Developers pick one model and over-provision everything — wasting 50–90% of compute

Meet LocalForge

A self-hosted AI control plane that orchestrates your local models intelligently



Works with:

- LangChain
- AutoGen
- CrewAI
- LlamaIndex
- Any OpenAI SDK

Built on 4 Technical Pillars:

- 🔍 Discovery
- 🧭 Routing
- 🧠 Memory
- 🔧 Finetuning + RAG

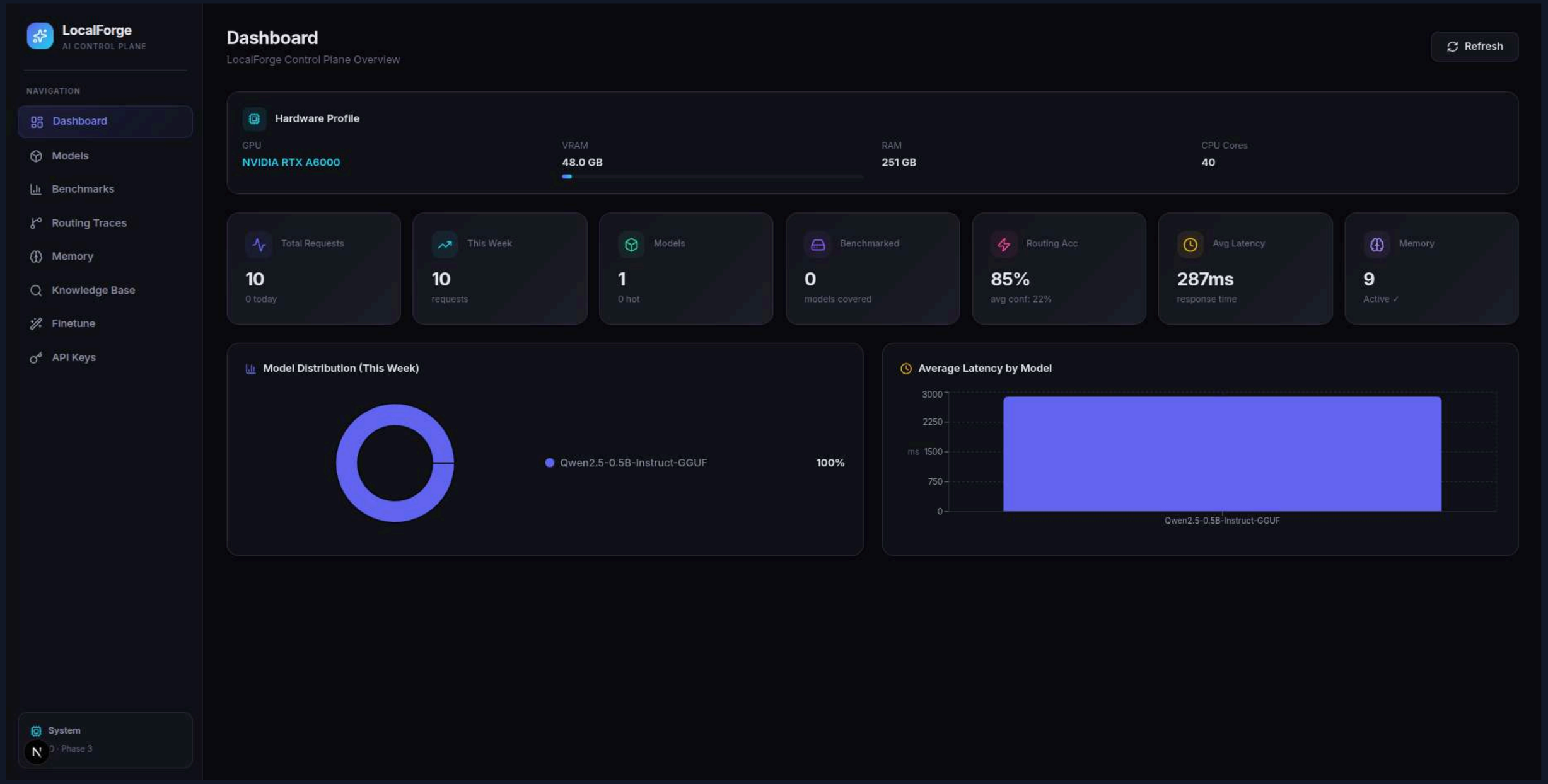
Pillar 1: Hardware-Aware Discovery

Only shows models that actually fit your GPU

Model Lifecycle State Machine

- One model HOT at a time — prevents VRAM crash
- Request queue during model swaps — no dropped calls
- Resident model stays loaded based on 24hr usage
- Finetuning lock — router excludes model during training

Pillar 1: Hardware-Aware Discovery



Pillar 2: Intelligent Router

Multi-signal routing — benchmarks + memory + latency + feedback

Final Scoring Formula

$$\text{final\_score} = 40\% \times \text{benchmark\_score} + 30\% \times \text{memory\_success} + 15\% \times \text{latency\_score} + 15\% \times \text{feedback}$$

with exponential decay $\lambda = 0.95$ on historical signals — recent interactions weighted higher

Query Classification — 5 Task Types:

🖥️ **Coding** | 📐 **Math** | 💡 **Reasoning** | 📋 **Instruction** | 💬 **General**

Benchmark Sources: MMLU-Pro • HumanEval • GSM8K • MT-Bench • GPQA-Diamond — pulled from HuggingFace Leaderboard API

LocalForge
AI CONTROL PLANE

NAVIGATION

- Dashboard
- Models
- Benchmarks
- Routing Traces
- Memory
- Knowledge Base
- Finetune
- API Keys

Routing Traces

Inspect every routing decision — query, scores, winner

What are good examples of model architectures used in multi-modal tasks?
general CONF: 23% → Qwen2.5-0.5B-Instruct-GGUF 7441ms

What is usefulness of the ml models like what are they and how we should use them later on
general CONF: 25% → Qwen2.5-0.5B-Instruct-GGUF 4991ms

What is the structure of ML models used in goat_integration?
coding CONF: 17% → Qwen2.5-0.5B-Instruct-GGUF 5768ms

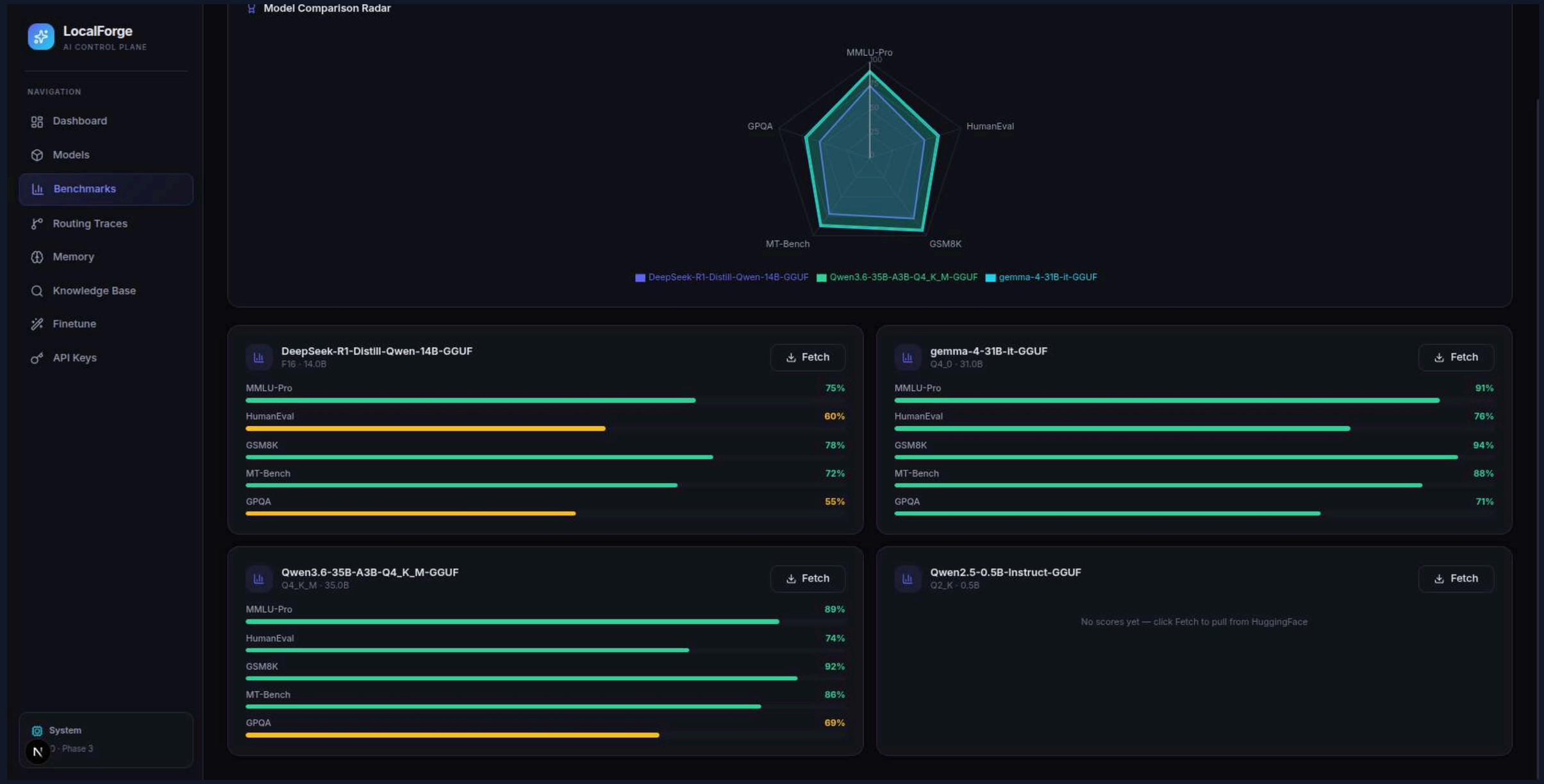
What is the structure of ML models used in goat_integration?
coding CONF: 17% → Qwen2.5-0.5B-Instruct-GGUF 875ms

Write a short haiku about coding.
instruction CONF: 25% MEMORY → Qwen2.5-0.5B-Instruct-GGUF 1644ms

According to the company policy, what is the remote work policy?
reasoning CONF: 20% MEMORY → Qwen2.5-0.5B-Instruct-GGUF 4237ms

According to the company policy, what is the remote work policy?
reasoning CONF: 20% → Qwen2.5-0.5B-Instruct-GGUF 1138ms

Real Benchmark Data — Per Model



Pillar 3: Memory-Enhanced Routing

A router that learns from your real workload — not just synthetic benchmarks

9	89%	4	ACTIVE
Stored Interactions	Weighted Success Rate	Feedback Signals	Memory Status

How It Works

1 — Embed Query

nomic-embed-text runs locally — 384d vector in <20ms

2 — Qdrant Search

Find top-K similar past queries from local vector DB

3 — Compute Score

Recency-weighted success rate per model — λ=0.95 decay

4 — Route

Best model selected — outcome logged back to memory

Routes tagged with [MEMORY] badge in dashboard when memory influenced the routing decision

Pillar 4: Finetuning + RAG

Train specialist models — augment with your private documents

FINETUNING PIPELINE

📁 Upload CSV / JSONL dataset

✅ Validate format and token distribution

⚙️ Configure LoRA rank, learning rate, epochs

📈 Live loss curve via Server-Sent Events

🔗 Auto-merge LoRA + export to GGUF

🚀 Auto-register into router — 0 clicks

Powered by Unsloth • 2x faster • 60% less VRAM • RTX 3080 compatible

RAG — KNOWLEDGE BASE LAYER

📄 **Upload any PDF, TXT, Markdown**

Chunked and embedded locally using nomic-embed-text

🔍 **Semantic retrieval per query**

Qdrant vector search finds top-K relevant chunks

🎯 **Router-aware KB selection**

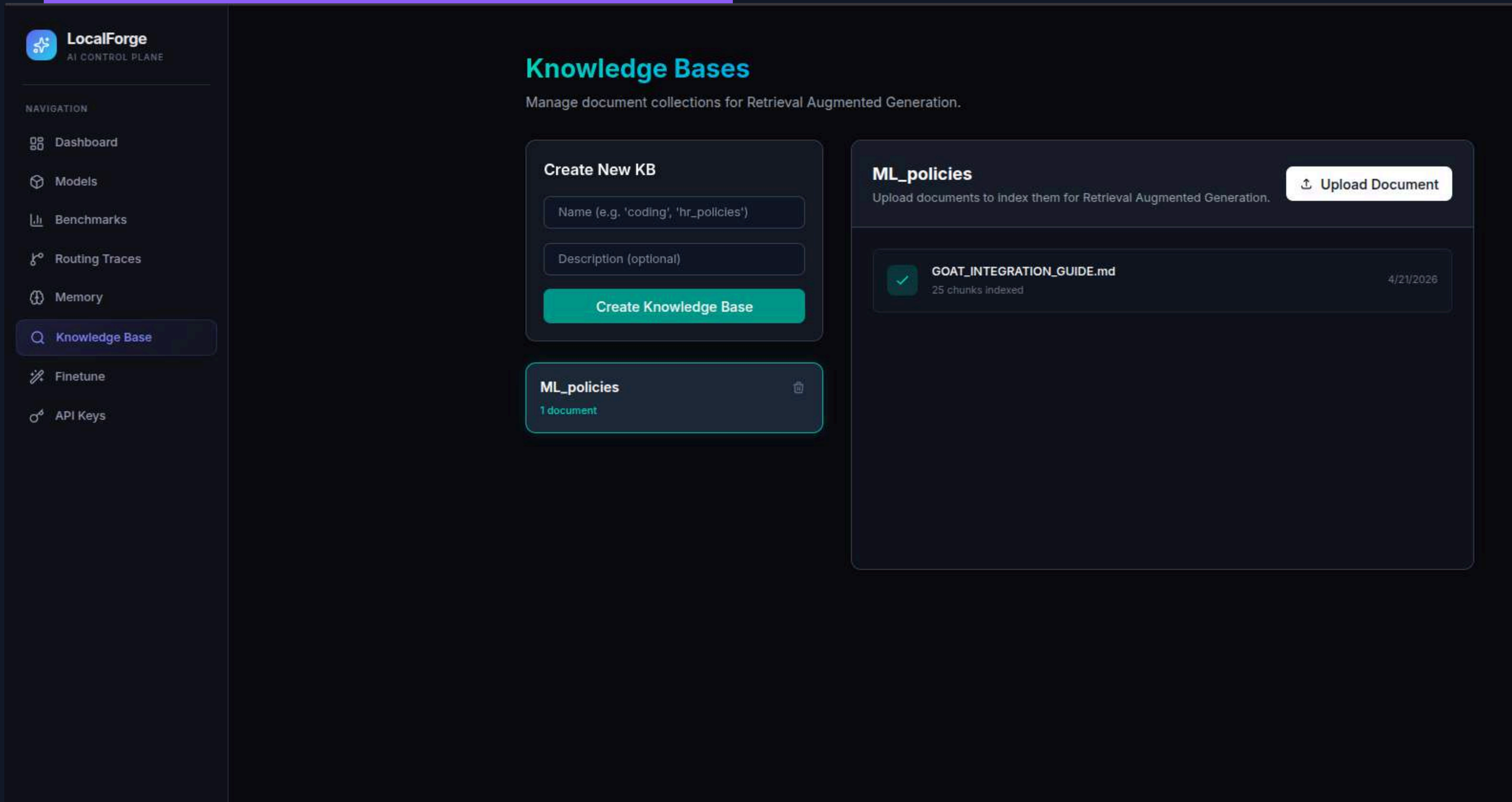
Right model + right knowledge base per query type

💉 **Context injection**

Chunks injected into system prompt before model call

Works with same Qdrant instance as Memory Layer — zero extra dependencies

RAG — KNOWLEDGE BASE LAYER



LocalForge
AI CONTROL PLANE

NAVIGATION

- Dashboard
- Models
- Benchmarks
- Routing Traces
- Memory
- Knowledge Base
- Finetune
- API Keys

System
Phase 3

Finetune

LoRA finetuning with automatic GGUF export & router registration

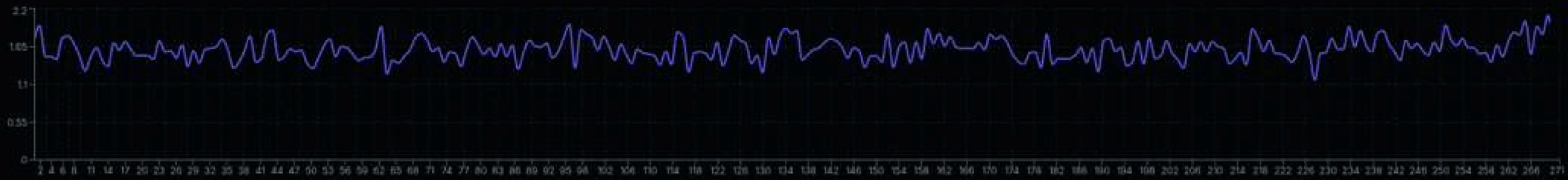
PEFT+TRL | Refresh

New Job | History (13)

Training in Progress

Cancel

PROGRESS	STEP	LOSS	STATUS
0%	271/69876	1.4124	training



Training started (3 epochs, ~69876 steps)

Demo: Using api key in contine.dev

CONTINUE

hi

Hello! How can I help you today?

So task is to create a flappy bird game in html

Thought for 27.5s

The user wants to create a Flappy Bird game in HTML. I should provide a complete solution using a single `index.html` file that contains the HTML structure, CSS for styling, and JavaScript for the game logic.

I will use the `create_new_file` tool to generate this file. The content needs to include:

1. **HTML**: A canvas element.
2. **CSS**: Centering the canvas, background color.
3. **JavaScript**:
   - Game loop (`requestAnimationFrame`).
   - Bird object (gravity, jump mechanics).
   - Pipe generation and movement.
   - Collision detection.
   - Score tracking.

I'll write the code directly into a file named `flappy-bird.html`.

Plan:

1. Define the HTML structure with `<canvas>`.

I'll create a single HTML file containing the structure, styles, and JavaScript logic for a Flappy Bird game. I will use an HTML5 Canvas element for rendering the graphics.

index.html — Create file

```
<!DOCTYPE html>
<html lang="en">
```

Local Config

Ask a follow-up

Welcome | config.yaml | index.html

home > ali_maadi > .continue > config.yaml

View Continue Reference | Download YAML extension for Intellisense

```yaml
name: Local Config
version: 1.0.0
schema: v1
models:
  - name: LocalForge (Auto)
    provider: openai
    model: auto
    apiBase: http://127.0.0.1:8010/v1
    apiKey: lf-b72d9709b4406e2a3da2194d33911216
```

Tech Stack & What's Next

Backend	Python 3.12 + FastAPI
Frontend	Next.js 16 + TypeScript + Tailwind + shadcn/ui
Local Inference	llama-cpp-python + llama.cpp server pool
Model Discovery	HuggingFace Hub API + Open LLM Leaderboard API
Finetuning	Unsloth + LoRA + GGUF export
Vector DB	Qdrant (local mode) — memory + RAG
Embeddings	nomic-embed-text — runs 100% locally
Storage	SQLite WAL — model registry, traces, keys

Roadmap

- Multi-node clustering
- RLHF pipeline
- MCP server integration
- Model distillation

LocalForge

Self-hosted • $0 per query • 100% data privacy • Drop-in OpenAI replacement • MIT License

THANK YOU!