Portkey AI Builders Challenge

Build Real Production AI Systems

Welcome to the Portkey AI Builders Challenge — a 24-hour hackathon focused on building real, production-grade AI systems, not demos.

This is not about chatbots or flashy UI.

This is about living systems that run continuously, manage state, adapt to change, and make correctness trade-offs.

If your system ran unattended for 6 months:

- would you trust it?
- would an enterprise trust it?

That's the bar.

🧩 Format

- Duration: 24 hours
- Team size: 3 people
- Teams: ~20
- Tracks: Open choice (pick any one problem below)
- Data: Real or synthetic
- Models & Tools: No restrictions

🤖 Embrace AI (Important)

We are actively looking for teams that leverage LLMs and AI tools deeply and creatively.

- There is no restriction on:
    - models (OSS or hosted)
    - providers
    - fine-tuning approaches
    - agent frameworks
    - developer AI tools
- You are encouraged to:
    - use LLMs as planners, critics, verifiers, and agents
    - use AI to write code, generate data, validate outputs, and reason about trade-offs
    - automate workflows using AI wherever it makes sense

This challenge is not about “minimising AI usage.”

It’s about using AI well.

🎯 What We’re Looking For

Across all projects, judges will look for:

- Continuous systems (not one-shot scripts)
- Thoughtful use of LLMs and AI tools
- State management & freshness logic

- Clear trade-offs (cost vs quality vs safety)
- Failure handling
- Explainability & observability
- Engineering rigor over UI polish

🚀 Challenge Tracks

Teams may pick any one of the following problems.

1️⃣ Living LLM Pricing & Capability Catalog

Problem

LLM providers constantly change:

- pricing
- context limits
- supported features
- rate limits
- model availability

These changes are fragmented across docs, blogs, APIs, and changelogs — and often not announced clearly.

Goal

Build a living system that continuously tracks LLM providers and models, and maintains a canonical catalog of:

- models
- pricing
- limits
- capabilities
- metadata

Core Requirements

Your system should:

- use agents to ingest data (scraping, APIs, feeds)
- maintain state to avoid unnecessary re-crawls
- detect semantic changes, not just text diffs
- attach metadata to every field:
    - source(s)
    - last verified timestamp
    - confidence score
- surface conflicting information instead of overwriting

Stretch Ideas

- pricing change alerts
- historical pricing deltas
- breaking-change detection
- machine-readable registry (JSON)

2️⃣ Smart Prompt Parser & Canonicalisation Engine

Problem

AI systems accumulate thousands of prompts:

- duplicated
- slightly modified
- hard to reason about
- impossible to optimise at scale

Goal

Build a system that analyses a collection of prompts and:

1. clusters semantically equivalent prompts
2. extracts prompt templates:
   - constant parts
   - variable slots
3. detects evolution:
   - brand-new prompt families
   - new variable sets in existing families

Requirements

- semantic similarity (not string matching)
- explainable grouping decisions
- incremental processing

- confidence thresholds for merges

3️⃣ Fine-Tuned OSS Guardrail Models

Problem

Prompt-based guardrails are:

- expensive
- slow
- inconsistent

Goal

Fine-tune one or more open-source models to act as specialised guardrail models.

Use Cases

Teams may fine-tune for one or more of:

- PII detection
- prompt injection detection
- content moderation
- policy violation detection
- unsafe instruction detection

Requirements

- synthetic data generation strategy
- clear train / eval split

- precision-focused evaluation
- runtime inference demo
- discussion of failure cases

Depth > breadth.

4️⃣ Cost–Quality Optimisation via Historical Replay

Problem

Model choices are often made blindly.

Goal

Build a system that:

- replays historical prompt–completion data
- evaluates across models and guardrails
- measures cost, quality, refusal rates
- recommends better trade-offs

Expected Output

🧠 Judging Criteria

Projects will be evaluated on:

1. Production readiness
2. Thoughtful use of LLMs & AI tools

3. System design & state management
4. Correctness & trade-offs
5. Engineering quality
6. Failure handling
7. Explainability

UI polish is optional.

Clear thinking is not.

🔗 Getting Started with Portkey (Recommended)

Before you start building, we strongly recommend that you:

- Go through the Portkey documentation
- Sign up on the Portkey platform
- Play around with the Gateway, models, and configurations

A lot of the things in this challenge — routing across models, guardrails, retries, logging, cost tracking, and experimentation — become significantly easier once you understand how Portkey works.

Additionally, by signing up you will:

- get AI credits across different model providers
- be able to experiment with multiple models quickly
- avoid re-building common plumbing from scratch

📌 Final Note

This is a builders challenge.

Use AI aggressively.

Automate ruthlessly.

Design systems that survive contact with reality.

Good luck — build something you’d actually ship.