Idea 2 Hackathon Pack (Updated) Probabilistic Failure Simulator + A/B Testing via Prompt Suites

Boilerplate + work assignment + 48-hour milestone calendar for a weekend build. Updated concept: run a **probe** over a suite of prompts (synthetic or real) to measure tokens/latency/tool calls, then apply deterministic failure rules and report **probability distributions** of failure per configuration (A/B).

Core demo moment: Run Prompt Suite on Config A vs Config B → show failure-rate distributions + P(B safer) + a break-first timeline.

Sponsor fit: Gemini (probe + explainers) + v0 (UI) + n8n (export tasks) + Cursor (build speed).

Executive Summary

- **Why this is stronger than sliders-only:** The simulator is grounded in *measured telemetry* from real prompt runs (prompt_tokens, retrieved_tokens, completion_tokens, latency_ms, tool_calls).
- **What we output:** For each config, estimate P(failure) across a prompt suite and show uncertainty (bootstrap CI or Beta posterior).
- **What “failure” means:** deterministic events like overflow, silent-truncation risk, latency breach, cost breach, tool timeout, retrieval-noise risk.
- **What stays simple:** We do *not* score “truth.” We predict system breakage + tail risk.

1) Boilerplate you can bring (rules-safe)

- **Repo skeleton:** empty Next.js + TypeScript app with Tailwind + shadcn/ui installed (no idea-specific logic).
- **Generic UI components:** Card, Badge, Tabs, Slider, Select, Toggle, Table, Toast.
- **Generic utilities:** JSON schema validator (zod), API error wrapper, logger, local file cache helper.
- **Deploy template:** Vercel config + generic README template.
- **Prompt-suite scaffolding:** empty folder /data/prompts with placeholder JSON schema (no actual prompt suite content prefilled).

Do not pre-build the rule thresholds, probe runner, or probability charts. Those must be built on-site.

2) What must be built on-site (core submission)

- **Config A/B definition:** model, context window, top-k, chunk size, max output tokens, tools on/off.
- **Prompt suite:** 50–200 prompts (synthetic templates + variations) with family tags (short/long, tool-heavy, doc-grounded).
- **Probe runner:** run each prompt through config and log telemetry (tokens, latency, tool calls).
- **Deterministic rules:** convert telemetry into failure events + breaks-at thresholds.
- **Probability layer:** estimate P(failure) + uncertainty; compute P(A safer than B).
- **UI dashboard:** distributions + breakdown by failure mode and prompt family + break-first timeline.
- **Export:** JSON + Markdown report; optional n8n workflow to create tickets/tasks.

3) Minimal data schemas

Prompt suite record

```
{
"id": "p_001",
"family": "long_context",
"use_case": "legal_qa",
"prompt": "…",
"expects_tools": false,
"expects_citations": false
}
```

Probe telemetry record (per prompt × config)

```
{
"prompt_id": "p_001",
"config": "A",
"prompt_tokens": 1320,
"retrieved_tokens": 4200,
"completion_tokens": 380,
"latency_ms": 2150,
"tool_calls": 0,
"tool_timeouts": 0
}
```

Failure event record (derived deterministically)

```
{
"prompt_id": "p_001",
"config": "A",
"failure_mode": "silent_truncation_risk",
"severity": "MED",
"breaks_at": "top_k>6 or context_usage>0.85",
"signal": {"context_usage": 0.91}
}
```

4) Deterministic rule set (starter)

Failure mode	Trigger	Severity	Signal	Mitigation
Context overflow	tokens_in > context_window	HIGH	context_usage	lower top-k / summarize / shorten system promp
Silent truncation risk	context_usage > 0.85	MED	context_usage	token budget + show dropped context indicator
Latency breach	latency_ms > SLO	MED/HIGH	latency_ms	cache, reduce context, tool timeouts
Cost runaway	cost_per_day > budget	HIGH	cost_per_day	cache, lower max_output_tokens, cheaper mod
Tool timeout risk	tool_calls>0 & timeouts>0	HIGH	timeout_rate	retries/backoff + fallback
Retrieval noise risk	top_k high OR low similarity	MED	top_k/sim	lower k, rerank, filter sources

Note: Keep similarity-based retrieval-noise optional (you can approximate with k and chunk size if time).

5) Probability layer (weekend-friendly)

- **Binary event:** for each prompt run, compute F=1 if any HIGH failure triggered (or define F per failure mode).
- **Failure probability:** p■ = (# failures) / N over the prompt suite.
- **Uncertainty option A (Bootstrap):** resample prompts 1000× → distribution of p■ → 95% CI.
- **Uncertainty option B (Bayesian):** prior Beta(1,1); posterior Beta(1+k,1+N–k); show credible interval.
- **A/B win probability:** report P(p_A < p_B) using posterior samples (or bootstrap delta).

On stage sentence: “We don’t pick the best answer; we pick the configuration with the lowest failure probability.”

6) Work assignment & distribution

Team of 3 (recommended)

- **Person A (Runner + rules):** probe runner, telemetry logging, deterministic rules → failure events.
- **Person B (Probability + charts):** bootstrap/Beta posterior, A/B delta + win probability, distributions by family.
- **Person C (Frontend + story):** UI (v0 + shadcn), dashboard layout, exports, README, demo video.

Team of 2 (if needed)

- **Person A:** runner + rules + probability (keep UI minimal).
- **Person B:** UI + charts + exports + demo script.

7) 48-hour calendar with milestones

Saturday	Deliverable
T0–1h	Repo + deploy hello-world (Vercel). Decide data schema. Create empty prompt suite file.
1–3h	UI skeleton: config A/B cards + prompt-suite upload/select + results placeholders.
3–6h	Probe runner for 10 prompts (serial). Log telemetry to JSON. Add caching + replay.
6–8h	Implement 4–5 rules. Produce failure events + break-first timeline.
8–10h	Probability layer: compute p■ per config + CI (bootstrap or Beta).
10–12h	Milestone Day 1: A/B run over 30 prompts → distributions + timeline + cached demo output.

Sunday	Deliverable
T0–2h	Scale to 80–150 prompts (cached runs; optional synthetic generator).
2–4h	Dashboard polish: breakdown by failure mode + prompt family; add win probability.
4–6h	Export MD/JSON; optional n8n workflow ‘Create tickets from mitigations’.
6–7h	Freeze features; harden fallbacks (cached output mode).
7–8h	README + 2–3 min demo video; final deploy + submission.

8) Sponsor-tech placement (visible + low risk)

- **Gemini:** probe runs + (optional) short mitigation explanations per failure mode.
- **v0:** scaffold the UI (A/B config cards, dashboard layout, tables).
- **n8n:** one-click export → create tasks/tickets from mitigations (optional).
- **Cursor:** dev accelerator (mention in README).
- **OpenAI/MiniMax:** fallback model only (do not make it required).

9) 90-second demo script

- Pick Config A vs Config B (e.g., top-k=10 vs top-k=4).
- Select prompt suite (e.g., 60 synthetic prompts across 4 families).
- Run (or replay cached run). Show token/latency distributions first (telemetry).
- Show failure probability distribution + CI per config + P(B safer).
- Show breakdown by failure mode and the break-first timeline.
- Export report; optional n8n: create tasks from mitigations.

Always keep a replay mode: cached outputs guarantee the demo if rate-limited.