MVP Requirements

UI/FrontEnd:

1. Web App - **shadcn**
    1. Inputs:
        1. Model
        2. Config:
            1. Temperature (from 0 to 2)
            2. top-k (from _ to _)
            3. Context Window (tokens)
            4. Costs ($)
            5. Chunk Size (input number)
            6. Tools (on/off)
            7. Max tokens
        3. the same prompts (temp) - a fixed list of 200 prompts (all kinds of prompts = long/short, with/wtihout tool use)
    2. Output:
        1. dashboard - failure rate, failure confidence band, description

BackEnd:

1. variables are accounted

Task division:

1. Person A: Chris
2. Person B: Emil
3. Person C: Yufei

Probabilistic Failure Simulator Hackathon Work Plans

Brief overview: Three coordinated tracks build a simulator that runs probes, logs failures, computes probabilities, and visualizes results. Plans include roles, responsibilities, timelines, dependencies, files, success criteria, and handoffs.

Person A: Runner + Rules Engineer

Role summary: Build probe runner, telemetry, and deterministic rules to generate failure events and break-first timeline.
Core responsibilities:

- Implement probe execution against configs.
- Log tokens, latency, tool calls.
- Apply rules to emit failure events and break-first timeline.
  Key files/folders:
- /lib/probe-runner.ts
- /lib/rules-engine.ts
- /lib/telemetry-logger.ts
  Function signatures:
- runProbe(config: ProbeConfig): ProbeResult
- loadConfigs(dir: string): ProbeConfig[]
- logTelemetry(t: TelemetryRecord): void
- evaluateRules(r: ProbeResult, rules: Rule[]): FailureEvent[]
- buildBreakFirstTimeline(events: FailureEvent[]): Timeline
  Dependencies:
- Inputs: /configs/*.json
- Outputs consumed by B and C.
  Handoff points:
- failure-events.json, break-first-timeline.json, telemetry.log
  Success criteria:
- Deterministic, reproducible events.
- Telemetry completeness (>98% probes logged).
- Break-first timeline aligns with events.
  Saturday timeline (hour-by-hour):
- H0–1: Repo setup, shared schema sync (all).
- H2: Scaffolding files; ProbeConfig schema (/types.ts).
- H3: loadConfigs(), runProbe() stub; sample configs.
- H4: logTelemetry() with token/latency/tool fields; write telemetry.log.
- H5: Implement evaluateRules() with rule DSL.
- H6: Emit failure-events.json {probeId, configId, mode, ts}.
- H7: buildBreakFirstTimeline(); write break-first-timeline.json.
- H8: CLI: npm run probes -> outputs all files.
- H9: Add seed for determinism; doc handoff format.

- H10–12: Milestone: freeze v1 outputs and schemas; push.
  Sunday timeline:
- H1: Bugfix runner edge cases; add retries.
- H2: Add more failure modes; rule examples.
- H3: Perf pass; batch I/O.
- H4: Schema lock; write README section for A.
- H5: Validate with B; sample datasets.
- H6: Feature freeze (all).
- H7–8: Final submission packaging.

Person B: Probability + Analytics Engineer

Role summary: Compute per-config failure probabilities, uncertainty, safer comparisons, and distributions.
Core responsibilities:

- $\hat{p}$ per config; 95% CI (Bootstrap/Bayesian).
- P(A safer than B).
- Distributions by failure mode/prompt family.
  Key files/folders:
- /lib/probability.ts
- /lib/statistics.ts
- /lib/analysis.ts
  Function signatures:
- estimatePhat(events: FailureEvent[], configId: string): {k: number, n: number, phat: number}
- bootstrapCI(k: number, n: number, alpha = 0.05): [number, number]
- bayesianBetaCI(k: number, n: number, alpha = 0.05): [number, number]
- compareConfigs(a: Stats, b: Stats): {pASafer: number}
- modeDistributions(events: FailureEvent[]): Record
  Dependencies:
- Consumes: failure-events.json, telemetry.log.
- Provides to C.
  Handoff points:
- analysis.json, distributions.json, comparisons.json
  Success criteria:
- Correct CIs; unit tests for edge cases.
- Clear safer-than metrics.
- Fast (<1s per dataset).
  Saturday timeline:
- H0–1: Schema sync (all).
- H2: Parse events; define Stats type.
- H3: estimatePhat(); write analysis.json (per config).
- H4: bootstrapCI(), bayesianBetaCI().
- H5: compareConfigs(); write comparisons.json.
- H6: modeDistributions(); write distributions.json.
- H7: Family aggregation; prompt-family.json.
- H8: Tests (/tests/statistics.test.ts).
- H9: Perf/validation against synthetic data.

- H10–12: Milestone: freeze API and outputs.
  Sunday timeline:
- H1: Edge cases (zero events); fallbacks.
- H2: Add CLI: npm run analyze.
- H3: Docs of JSON schemas.
- H4: Align with C on visualization shapes.
- H5: Final QA with A datasets.
- H6: Feature freeze (all).
- H7–8: Submission artifacts.

Person C: Frontend + Story Engineer

Role summary: Build UI to input configs, show analytics, timelines, breakdowns; export; demo and README.
Core responsibilities:

- Scaffold app with v0, shadcn/ui.
- Cards for Config A/B.
- Dashboard for distributions and timelines.
- Export JSON/Markdown; optional n8n.
  Key files/folders:
- /app
- /components
- README.md
- /public/demo
  Components and functions:
- ConfigCard.tsx
- Dashboard.tsx
- TimelineChart.tsx
- DistributionChart.tsx
- loadData(files: string[]): AppData
- exportJSON(data: AppData): File
- exportMarkdown(summary: Summary): File
  Dependencies:
- Consumes A/B JSONs.
- Requires stable schemas from B.
  Handoff points:
- UI consumes analysis.json, comparisons.json, distributions.json, break-first-timeline.json
  Success criteria:
- Clear comparisons and timelines.
- Export works.
- Demo video shows end-to-end.
  Saturday timeline:
- H0–1: Design and data schema sync (all).
- H2: Next.js + shadcn scaffold (/app).
- H3: ConfigCard.tsx; file picker.
- H4: Data loader; schema validation.
- H5: Dashboard layout; cards/sections.
- H6: TimelineChart wired to break-first-timeline.json.

- H7: DistributionChart from distributions.json.
- H8: Safer-than widget using comparisons.json.
- H9: Export JSON/MD.
- H10–12: Milestone: UI v1 walkthrough.
  Sunday timeline:
- H1: Polish UI; accessibility.
- H2: n8n integration stub (optional).
- H3: README.md authoring; usage.
- H4: Record demo video.
- H5: Bugfixes; perf.
- H6: Feature freeze (all).
- H7–8: Final submission packaging.