PR Risk & Velocity Copilot (No RAG Version)

Objectives (1 week)

Build an AI agent that:

1. Answers engineering metrics questions from NL queries

2. Reviews a given PR for risk and outputs a risk score + actionable comments
   All grounding must come from the database and hard-coded/checked rules—no RAG.
3. Set up **Langsmith** integration to enable agent tracing, tool tracking, and execution time logging.

Data & Minimal Schema

Same as before (no doc stores). Tables like:

- `pull_requests`, `reviews`, `commits`, `ci_runs`, `coverage_reports`, `ownership`, `hotspots` (optional)

Derived metrics:

- Cycle time, first-response latency, churn per PR, coverage deltas, etc.

MCP Server (Required Tools)

Expose JSON tools; block non-SELECT SQL:

1. `list_tables()` → `{tables:[{name,row_count}]}`

2. `get_related_tables(table)` → FK relations

3. `run_query(sql, params)` → row-capped, paginated

4. `get_metric(metric_name, window, filters)` → typed helpers (cycle_time, review_latency, churn, …)
5. `get_pr_summary(pr_id)` → PR + CI + coverage deltas
6. `get_diff_outline(pr_id)` → changed files + ownership + optional hotspots
7. `safe_sql(plan, schema_guard=true)` → validate columns/tables; auto-LIMIT; deny DDL/DML

Guardrails: deny destructive ops, cap rows, return typed errors & explainers.

Agent Flow (LangGraph-style)

1. **Intent Classifier** → `MetricsQuery` | `PRRiskReview`
2. **Query Planner** → plans required metrics & SQL sketches (no RAG)
3. **Schema Confirm** → `list_tables`, `get_related_tables`
4. **SQL Synthesis & Execute** → `safe_sql` → `run_query`
5. **Validator** → shape checks, nulls/outliers; re-plan on anomalies
6. **Risk Heuristics** → combine `get_pr_summary` + `get_diff_outline` with rules:
    - Large diff (e.g., >1500 LOC or >30 files) → raise risk
    - Ownership mismatch in core modules → raise risk
    - Coverage drop (<0) → add test action
    - First-response latency > N hours → flag bottleneck
    - Optional hotspots/churn thresholds
      Output: **risk score (0–1)** + **exactly 3 actionable comments**
7. **Answer Composer** → tables + short narrative + assumptions

Required Demo Queries

Metrics

- “Rank teams by median PR cycle time (last 30 days). Show p50/p75 + PR count.”
- “Top 10 PRs by review latency this week and who first responded.”
- “Is backend churn above the threshold we set? Show current vs threshold.”

PR Risk

- “Review PR #1245: risk score + 3 actionable comments.”
- “List files in PR #1245 lacking primary team ownership.”

Explainability

- “How did you compute cycle time?” → show SQL expressions used.

Deliverables

1. **MCP server** (code + tool docs)
2. **Agent app** (graph, prompts, tool bindings)
3. **Seed DB & migrations** (+ generator for synthetic data)
4. **Evals** (metrics queries + PR risk cases; assertions/tolerances)
5. **Demo script** (5–6 canned prompts)
6. **Observability** (Langsmith integration and structured logs)

Required Demo Queries (Must Work)

Metrics

1. “Rank teams by median PR cycle time (last 30 days). Show p50/p75 and PR count.”
2. “Top 10 PRs by review latency this week with the reviewer who first responded.”
3. “Is backend churn above 20 this sprint? Show current vs threshold.”

PR Risk

4. “Review PR #1245 and give a risk score and exactly 3 actionable comments.”
5. “List files in PR #1245 that lack primary team ownership.”

Explainability

6. “How did you compute cycle time? Show the SQL expression used.”

Evaluation (100 pts)

- MCP server quality (20): typed I/O, pagination, errors, guards
- Agentic behavior (20): tool choice, re-plans, retries, uncertainty handling
- SQL synthesis & validation (20): correct grouping/filters/limits, schema checks
- Risk review quality (20): meaningful score + specific, testable actions
- Observability (10): Langsmith logs of plans, SQL, durations, tool calls
- DX & Docs (10): clear README, diagrams, run scripts