

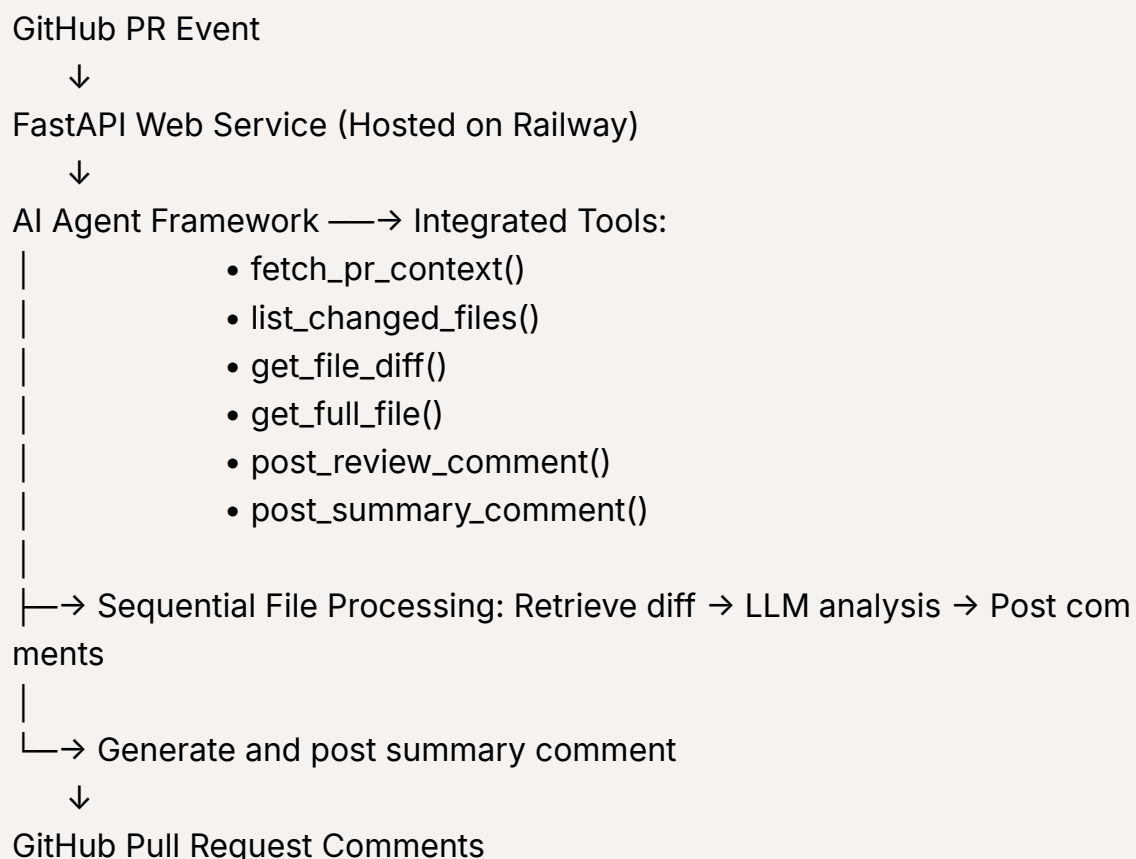
# Idea 2 → AI Code Review Agent

## 1. Project Objective

This automated GitHub integration uses artificial intelligence to perform code reviews on pull requests. When a pull request is created, the system analyses the changed code and posts inline review comments directly to the GitHub interface. The solution uses a minimalist architecture, with no persistent database storage or dedicated frontend components.

## 2. System Architecture

The following diagram illustrates the high-level system architecture and data flow:



## 3. Technology Stack Evaluation

## 3.1 Implementation Options

Component	Google AI Stack	Anthropic/OpenAI Stack
Web Framework	FastAPI	FastAPI
Agent Framework	Google Agent Development Kit (ADK)	LangGraph
Language Model	Gemini 3 Pro	Claude Sonnet 4 / GPT-4
GitHub Integration	PyGithub	PyGithub
Logging Mechanism	Python Standard Logging	Python Standard Logging
Deployment Platform	Railway	Railway

## 3.2 Cost Analysis

- **Gemini 3 Pro:** \$0.04 per review
- **Claude Sonnet 4:** \$0.06 per review
- **GPT-4:** \$0.10 per review

## 4. Core Functional Components

### 4.1 PR Context Retrieval ( `fetch_pr_context` )

Obtains pull request metadata including title, description, author identifier, and assigned labels to establish review context and understand implementation intent.

### 4.2 Changed Files Enumeration ( `list_changed_files` )

Provides a comprehensive overview of all modified files, including file paths, programming languages, and extent of changes, enabling intelligent review prioritization.

### 4.3 File Diff Extraction ( `get_file_diff` )

Retrieves the specific differential content for an individual file. Token consumption typically ranges from 3,000 to 15,000 tokens per file analysis.

### 4.4 Full File Retrieval ( `get_full_file` )

Acquires the complete file content when differential data alone proves insufficient for comprehensive analysis. This operation is performed conditionally based on contextual requirements.

## 4.5 Inline Comment Publication ( `post_review_comment` )

Posts targeted review comments on specific code lines within the pull request, with full Markdown formatting support for enhanced readability.

## 4.6 Summary Comment Generation ( `post_summary_comment` )

Publishes a comprehensive assessment comment summarizing all identified issues, including quantitative metrics and overall recommendations.

---

# 5. Review Execution Strategy

## 5.1 Incremental Processing Workflow

The system implements a sequential file-by-file analysis approach to ensure consistent performance and resource management:

1. **Context Acquisition:** Retrieve pull request metadata and objectives
2. **File Enumeration:** Compile complete list of modified files
3. **Exclusion Filtering:** Omit dependency lock files, generated code artifacts, and distribution directories (e.g., `dist/`, `node_modules/` )
4. **Priority Determination:** Prioritize source code files, modifications with significant change volume, and newly introduced files
5. **Processing Limit:** Maximum of 10 files per review cycle to maintain optimal performance
6. **Per-File Analysis:**
  - Extract differential content for the individual file
  - Submit file context and PR metadata to the language model
  - Process LLM-generated analysis and identified issues
  - Publish review comments immediately upon generation
  - Log token utilization metrics for cost monitoring
7. **Final Summary:** Post comprehensive review summary upon completion of all file analyses

## 5.2 Rationale for Sequential Processing

The incremental approach provides several operational advantages:

- **Predictable Resource Consumption:** Approximately 5,000 tokens per file enables accurate cost forecasting
  - **Focused Analysis:** Isolated file examination promotes thorough and contextually relevant feedback
  - **Scalability:** Accommodates pull requests of arbitrary size without system degradation
  - **Controllable Termination:** Allows graceful interruption if token limits or cost thresholds are encountered
- 

## 6. Reference Documentation

### 6.1 Google AI Stack

- [Gemini API Documentation](#)
- [Agent Development Kit \(ADK\) Repository](#)
- [Google AI Studio Platform](#)

### 6.2 Anthropic/OpenAI Stack

- [Claude API Documentation](#)
- [LangGraph Framework Documentation](#)
- [Anthropic Developer Console](#)

### 6.3 Common Infrastructure

- [FastAPI Web Framework](#)
- [PyGithub Library Documentation](#)
- [Railway Deployment Platform](#)
- [GitHub Webhooks Specification](#)