CodeCoach

By: Ethan, Maria, Katherine, Sambhav, Tanisha

Table of contents

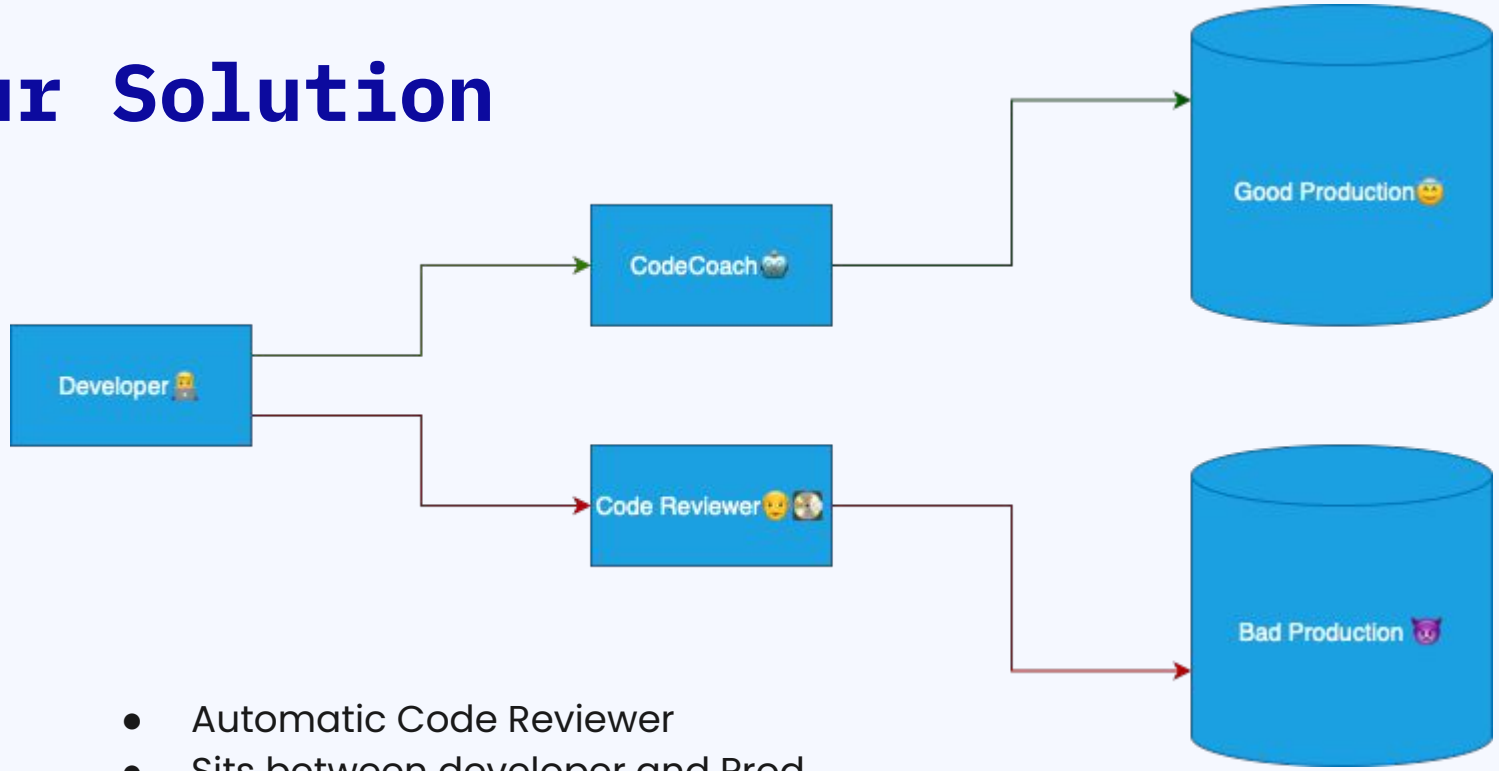
01

Opening Remarks

Code is Vulnerable

- Knight Capital lost 440 million in 45 mins
- Boeing Crash led to 346 people
- Messy software/poor reviewing can lead to disaster

Our Solution



- Automatic Code Reviewer
- Sits between developer and Prod
- Goal: Eventually get rid of the code reviewer while improving code safety

02

What does our App do

Live Demo

03

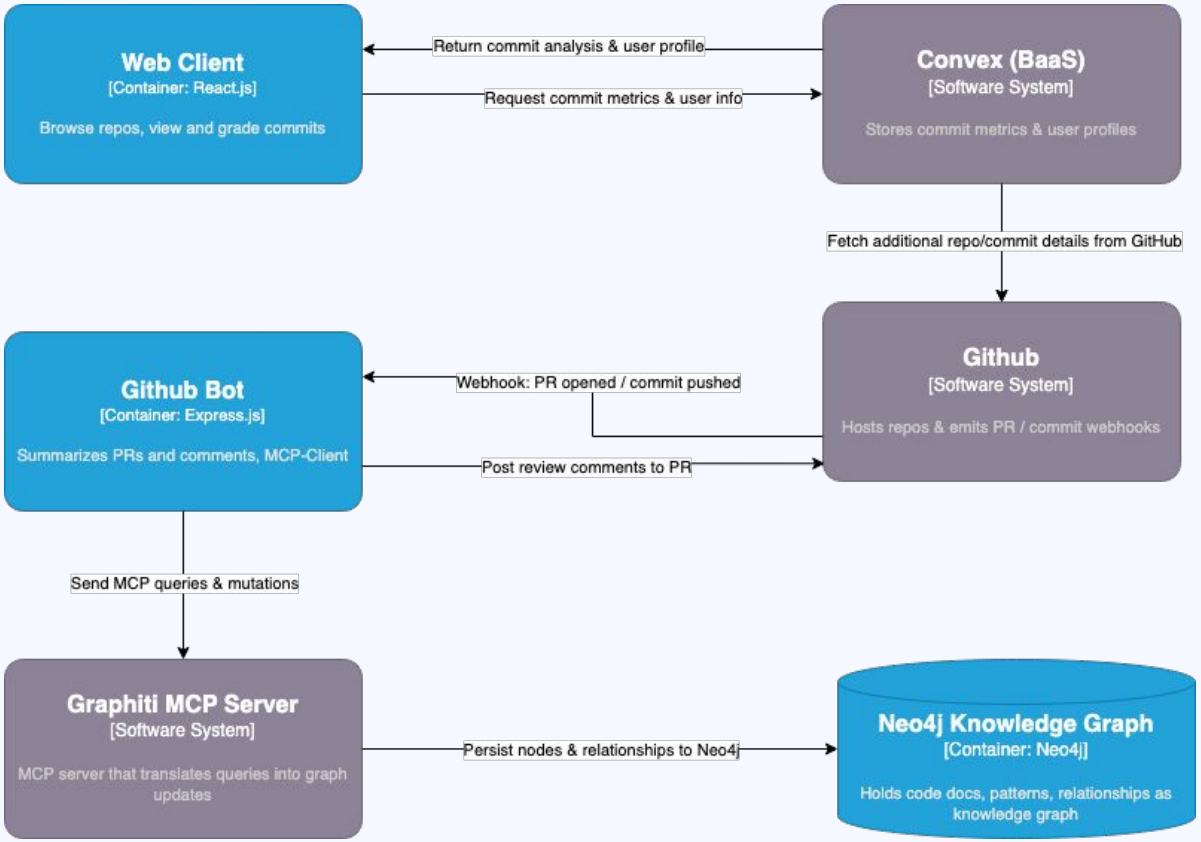
How our App Works

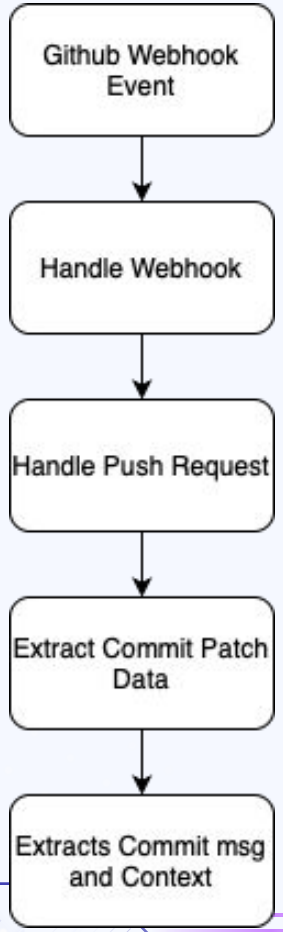
High-level Architecture: Sprint 1

- No database configured to the Github bot
- Why? We originally thought the bot simply reacts to GitHub webhooks and posts comments; it doesn't need to remember anything between events

High-level Architecture: Sprint 2

- We added Convex for a quick frontend demo
- Bot and Convex run separately; difficulty communicating





AI Grading

01 Detect Git Push

Using handleWebhook to receive the event

02 Extract Data

Using GitHub API to get metadata and changed files

```
Fetching repository node ID...
Fetched repository node ID for sam-chordia/OS_sandbox: R_kgDOONmZug
Upserting repository with node ID: R_kgDOONmZug
Upserting repository: sam-chordia/OS_sandbox with GitHub ID: R_kgDOONmZug
Looking up by provided GitHub ID: R_kgDOONmZug
Lookup result: { repositoryId: 'k17bytwqepbeq90g6zscn6bjkd7ep8we', status: 'found' }
Found existing repository with ID: k17bytwqepbeq90g6zscn6bjkd7ep8we
Repository ID from Convex: k17bytwqepbeq90g6zscn6bjkd7ep8we
Repository ID k17bytwqepbeq90g6zscn6bjkd7ep8we doesn't have proper Convex format
Found repository ID by direct lookup: k17bytwqepbeq90g6zscn6bjkd7ep8we
Storing commit 95c633e for repository k17bytwqepbeq90g6zscn6bjkd7ep8we with score 10
✅ Stored commit data in Convex for OS_sandbox
```

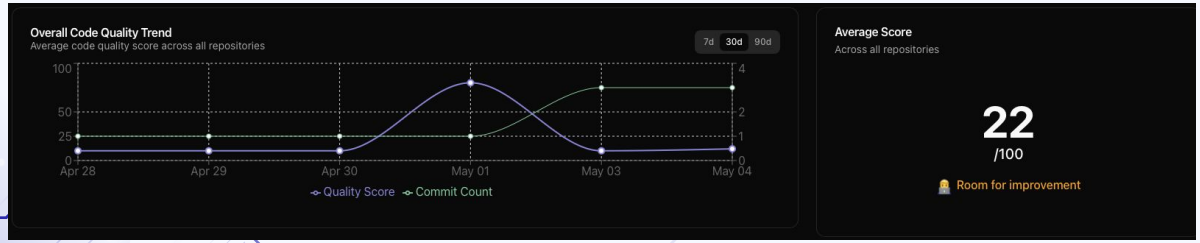
AI Grading

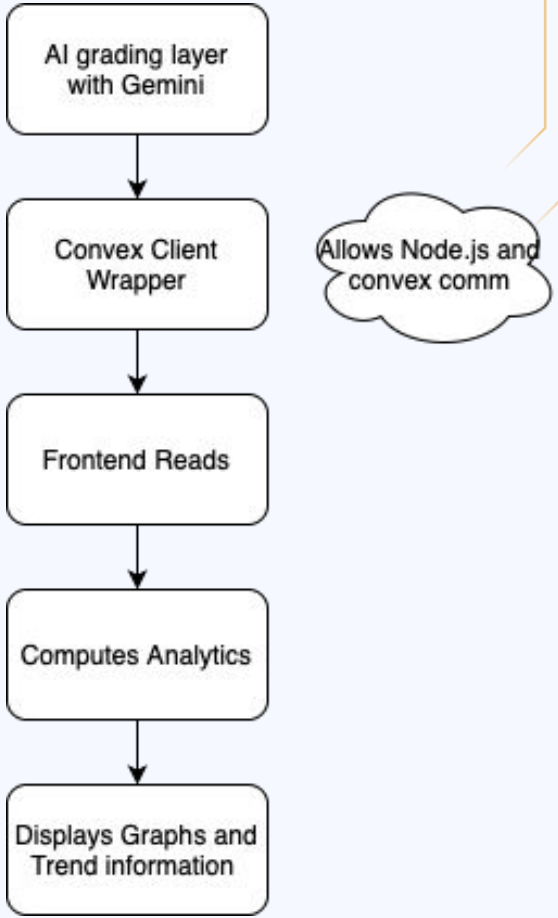
03 **Analyze Patch and Grade**

Using CommitGrader Agent

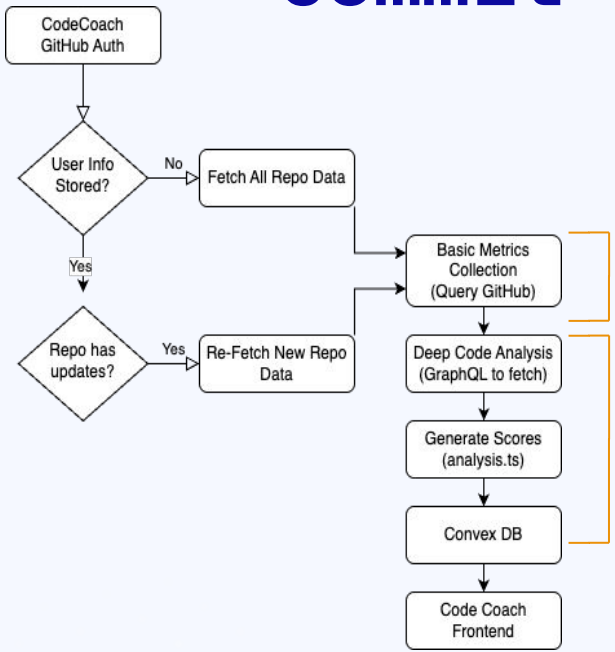
04 **Save Score**

Use Convex to store grades and frontend to display analysis





Commit Metrics + Code Smells



1 Repository Data Retrieval

When the user logs into CodeCoach:

- Their GitHub token is stored
- Convex fetches repos via GitHub's GraphQL API
- useGithubRepositories hook manages this process

2 Metrics & Smells Generation

In-depth code quality analysis performed by:

- Examining code structure and patterns
- Analyzing 15-30 files per repository
- Calculating complexity, duplication, test coverage
- Storing results in Convex database

```
// Calculate weighted score
const score =
  (adjustedTestCoverage * 0.2) + // 20% weight
  ((100 - duplication) * 0.2) +  // 20% weight
  (docCoverage * 0.25) +         // 25% weight
  (maintainabilityValue * 0.25) + // 25% weight
  (Math.max(0, 100 - securityIssues * 5) * 0.1); // 10% weight
```

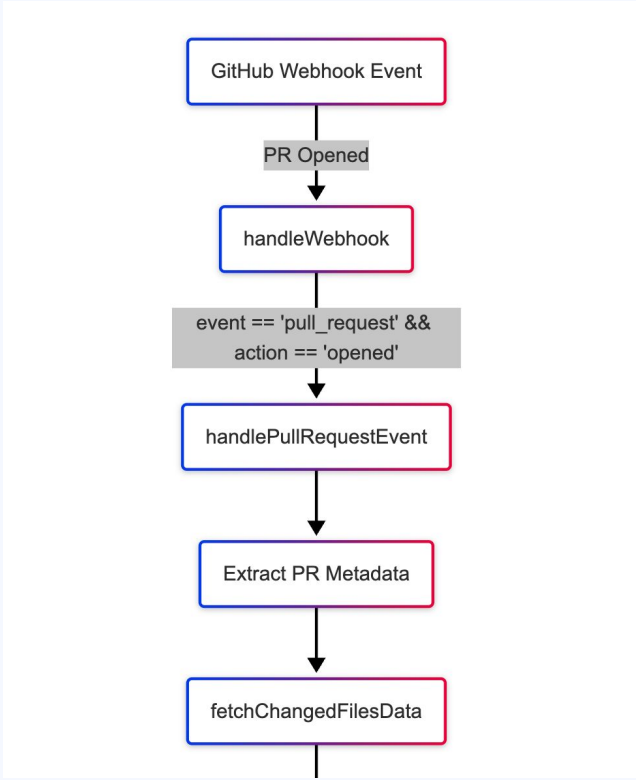
Github-bot PR Summarization

01 **Detect PR Opened**

Using handleWebhook to receive the event

02 **Extract Data**

Using GitHub API to get metadata and changed files



Github-bot PR Summarization

03 Generate Feedback

Using CodeReviewAgent to generate three different feedbacks

Commit Logic

consistency
+
purpose

Static Analysis

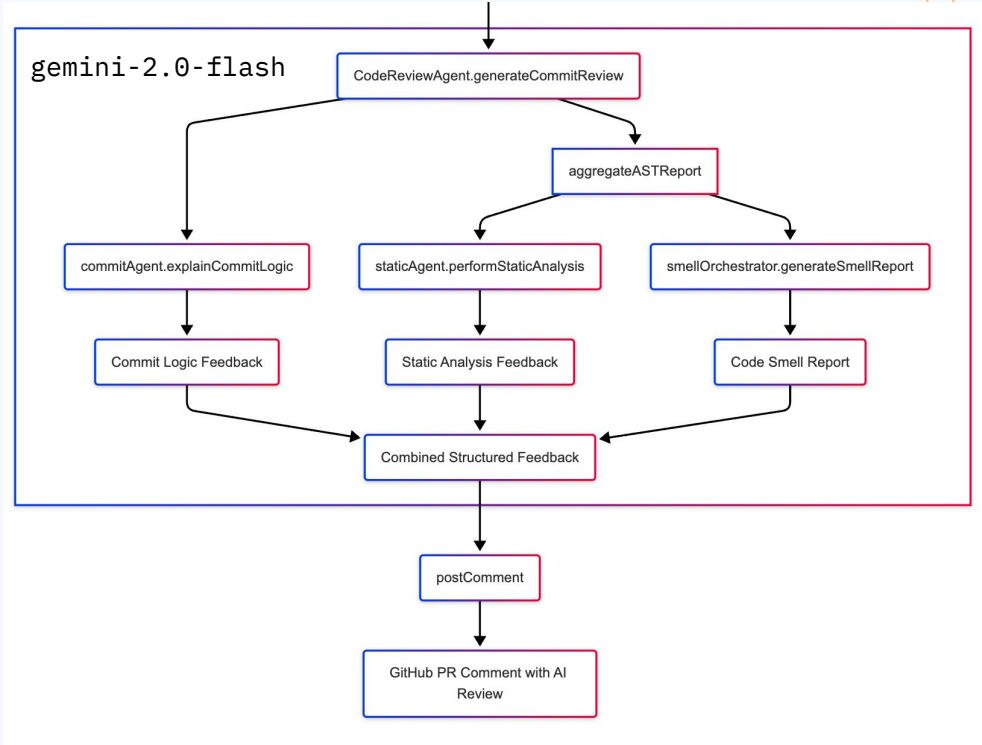
structure
+
bug

Code Smell

NamingConventionAgent
DuplicateCodeAgent
LargeFunctionAgent
…

04 Post Feedback

Using GitHub API to post PR comments



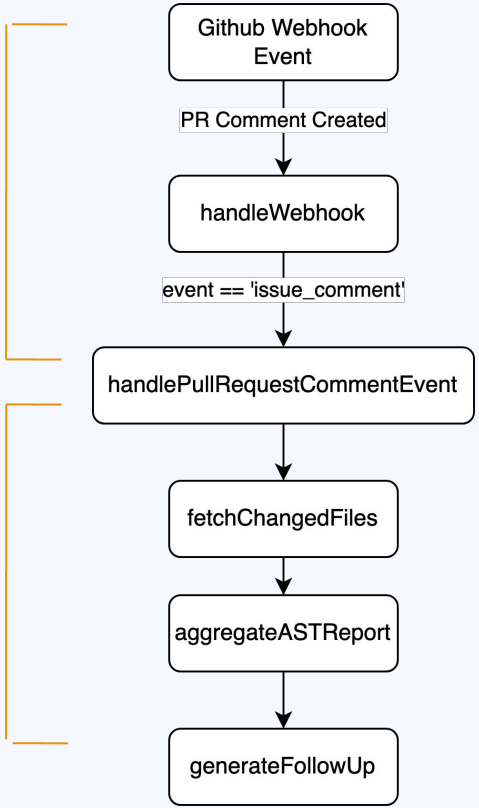
Github-Bot Chat Feature

01 Comment Created

Use handleWebhook to receive the Github webhook event

02 Extract Data

Adds previous comments as context for the follow-up response, as well as changed file data



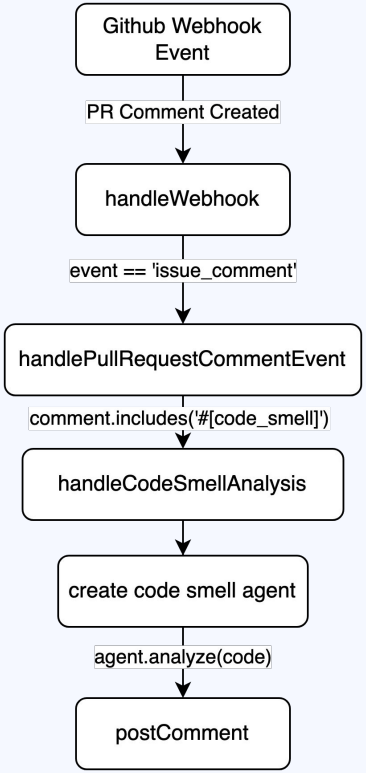
Github-Bot Chat Feature

03 Generate Feedback

Creation of specialized agents for 6 different code smells: naming, duplicate, largefunction, magicnumber, longparameter, complexity

04 Post Feedback

Use GitHub API to post PR comments



The Graphiti Journey:Overview

Graphiti is a Python library for building and querying dynamic, temporally aware knowledge graphs.

Knowledge graph is a dynamic network of entities and their relationships that provides LLMs with precise, structured context for retrieval-augmented generation.

01

Local Graphiti + Flask API

02

an MCP-compliant Agent

Extra Info

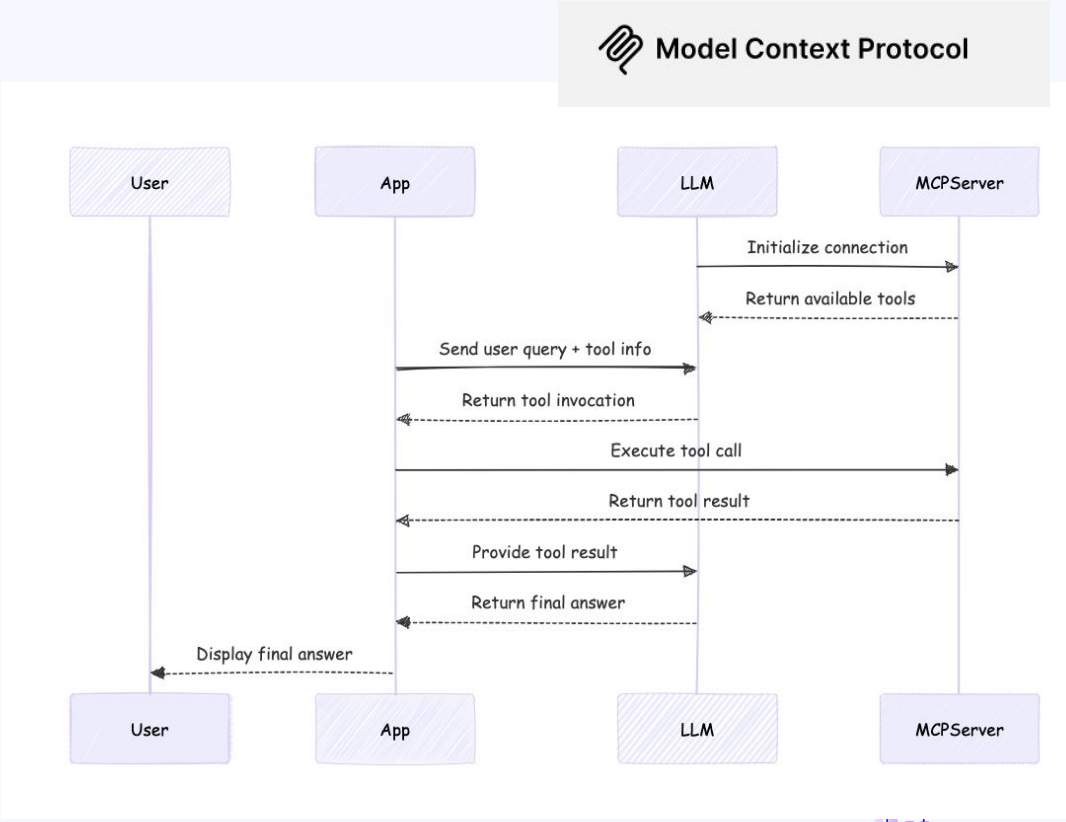
- Episodes = Raw Data
  - Each “episode” node holds the original text or JSON event you add, preserving full context.
- Entities = Concepts
  - Entity nodes capture the people, places, things, or ideas mentioned in episodes.
- Episodic Edges = Mentions
  - Whenever an episode mentions an entity, an edge links that episode node to the corresponding entity node.
- Entity Edges = Facts
  - Edges between two entity nodes store explicit relationships or properties (i.e., extracted facts).

https://blog.getzep.com/llm-rag-knowledge-graphs-faster-and-more-dynamic/

Making an Model Context Protocol (MCP) Client

"a USB-C port for AI applications"

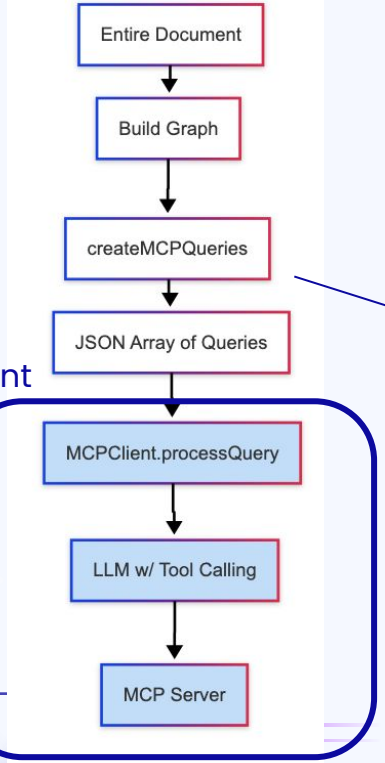
An API Layer
MCP provides a standardized way to connect AI models to different data sources and tools.



Credit:https://modelcontextprotocol.io/introduction
Credit: Alejandro AO - Software & Ai

Prompt Engineering w/ Graphiti

Graphiti Agent



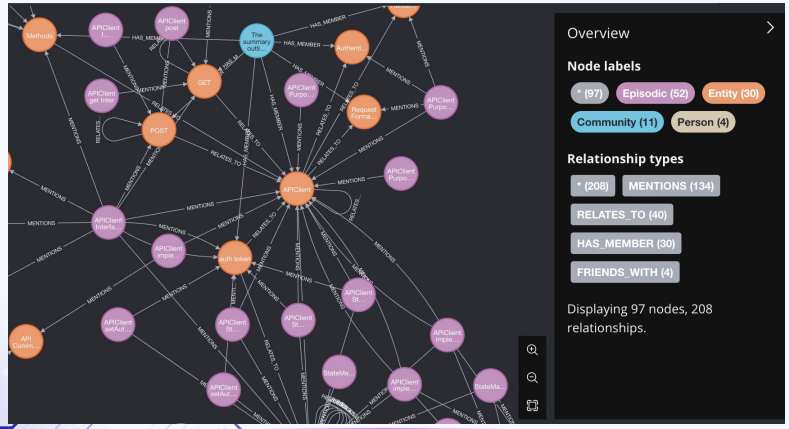
Smaller, more focused queries was more consistent and accurate than longer queries

Array of Queries:

```
  "clear_graph"
]
GraphitiAgent initialized successfully
  Running buildGraph on "inline-doc.md" with inline content…
Processing CodeCoach documentation: inline-doc.md
[
  "Add an episode with groupid . with name 'StateManager Purpose' and body 'Provides centralized state management for the appl
ocal and global state.'",
  "Add an episode with groupid . with name 'StateManager.createStore' and body 'Creates a new state store with the provided in
  "Add an episode with groupid . with name 'StateManager.useStore' and body 'Hook to access and subscribe to store state.'",
  "Add an episode with groupid . with name 'StateManager.dispatch' and body 'Method to update store state via actions.'",
  "Add an episode with groupid . with name 'StateManager Dependencies' and body 'Dependencies: EventEmitter, LocalStorage'",
  "Add an episode with groupid . with name 'APIClient Purpose' and body 'Handles all communication with backend APIs, includin
tion, and error handling.'",
  "Add an episode with groupid . with name 'APIClient.request' and body 'Makes a request to the specified endpoint.'",
  "Add an episode with groupid . with name 'APIClient.get' and body 'Shorthand for GET requests.'",
  "Add an episode with groupid . with name 'APIClient.post' and body 'Shorthand for POST requests.'",
  "Add an episode with groupid . with name 'APIClient.setAuthToken' and body 'Sets the authentication token for subsequent req
  "Add an episode with groupid . with name 'APIClient Dependencies' and body 'Dependencies: StateManager (for auth stat'"
]
{"message": "Episode 'StateManager Purpose' queued for processing (position: 1)"}
```

The Visualization

The **Knowledge Graph** visualization was created using a Neo4j JavaScript driver to connect to the database instance and Neo4j Visualization Library, NVL for short. NVL provides a way to visually represent data stored in Neo4j's graph database format.



02

Web Client Graph

04

Closing Remarks

The Limitations

1. Not deployed
2. Limited to 6 code smells
3. Lacks mergability reporting
4. Our AST report can only be generated with JavaScript
5. Graffiti chatbot is static on the frontend
6. Limited to 30 files when ingesting repository
7. AI agents do not currently use Graphiti knowledge graph
8. Feedback is purposefully limited to only changed, added, and deleted files

The future of your software

Long story short:
Redo it!

With more experience building a Github App, we know now a better software architecture

The CodeCoach Team