HW 3 v1

The Assignment

Overview

Up until now, you have built isolated components: a Chat service, a Ticketing service, or an AI service. In this final phase, you are not building new features from scratch. Instead, you are acting as a **Systems Integrator**.

Your goal is to assemble these components into a single, cohesive pipeline that allows a user in a chat interface to manage work tickets using natural language.

The final product must be able to:

- **Answer questions** by leveraging an AI model.
- **Fetch and summarize tickets** from a ticketing system (e.g., "Show me my 3 most recent open tickets").
- **Take action on tickets** (e.g., "Close this ticket").

User Flow

Here is the specific workflow your deployed application must support:

1. **Input:** User types a command in Chat (e.g., *"Create a ticket for fixing the login bug"*).
2. **Routing:** The Chat application sends this text to the AI Service.
3. **Reasoning:** The AI Service analyzes the text, determines the intent (`create_ticket`), extracts the data (`title="Fix login bug"`), and returns a structured tool call.
4. **Execution:** The application executes this call against the standardized Ticket Interface.
5. **Response:** The Ticket Service confirms the action, and the Chat Interface relays the success message back to the user.

Instructions

The core steps are:

1. **Integration:** Your primary task is to make three disparate systems—Chat, Tickets, and AI—communicate effectively.

2. **Deployment:** This is not a local-only project. You will deploy your application and manage its infrastructure as code.
3. **Observability:** Your deployed application must emit telemetry data to monitor its health and performance.

Interface Repository

IoC & Telemetry Your application must be deployed, and its infrastructure must be managed using an **Infrastructure as Code (IaC)** tool like Terraform or AWS CloudFormation. You are responsible for provisioning the necessary resources (e.g., servers, databases, environment variables) in a repeatable and automated way.

Furthermore, your deployed application must emit **telemetry data**. This is non-negotiable and critical for understanding the performance of a live service. You must implement monitoring for:

- **Request Latency:** How long does each API call or user interaction take?
- **Success Rate:** What percentage of requests are completed successfully?
- **Failure Rate:** What percentage of requests result in errors?

You should use a monitoring or observability platform to collect and visualize this data.

I recommend that you have a working version of this by the HW 3 Amended > Second Submission deadline

First submission By the turn of the month, your individual team's project must be refactored to implement the shared, standardized API for your vertical. This is a hard deadline, as other teams will be depending on this standardized interface for the next phase.

Second Submission Choose **one** of the other two verticals (either Tickets or AI) and integrate their system into your project. For example, if you are a Chat team, you might integrate an AI system first. Your submission must:

- Demonstrate successful integration with at least two different providers from that vertical (e.g., a Chat bot working with both OpenAI and Gemini).
- Include integration tests that verify the two systems are working together correctly.

I recommend you also have your IoC and Telemetry setup by now.

Final Submission This is the final deliverable. It includes your complete, three-part system, documentation, and a video demonstration.

- **Pull Request:** A clean, well-documented PR with your final code.

- **Video Demonstration:** You must record a video that includes:
  - A clear explanation, in your own words, of how your complete project works.
  - A live demonstration of its functionality, showcasing integration with different providers (e.g., swapping Jira for Trello).
  - A walkthrough of your CircleCI deployment pipeline.
  - An explanation of the tests being run on the cloud.
  - A high-level overview of your end-to-end (e2e) and integration tests and what they verify.
  - A view of your telemetry dashboard showing request latency and success/failure rates.

Deadlines

- **11/21:** Draft of this assignment is provided.
- **11/26:** Draft is frozen.
- **12/27 - Interface Deliverable:** Verticals must submit their agreed-upon shared interface memo and individual team alignment plans.
- **12/8 - First Submission:** All teams must have their code aligned with the shared API.
- **12/12 - Second Submission:** First cross-vertical integration is complete.
- **12/14:** Peer feedback and TA feedback on the first integration.
- **12/19 - Final Submission:** Final project, including the video demonstration, is submitted.