Sentellent Hiring Challenge: Build a "**Contextual Agentic AI Assistant"**

Deadline: January 24th, 11:59 PM
Role: Full Stack AI SDE Intern
Company: Sentellent

Submisson Link: forms.gle/VVd9r4kemkWoiQVQ8

The Vibe Check (Read This First)

We know we are living in the era of AI. **You can use ChatGPT, Claude, GitHub Copilot, or "vibe code" however you want.** We don't care if you write every line from scratch or if you orchestrate AI to do it for you.

However, here is the catch: In 2026, *writing* code is easy. **Deploying, maintaining, and automating** code is the real engineering challenge. We are placing **heavy weightage** on your **Deployment (AWS)** and **CI/CD pipelines**. A fancy AI agent that only runs on localhost:3000 is worth less to us than a simple agent that is perfectly containerized, terraformed, and live on the internet.

The Mission

Your task is to build a **Personal Agentic AI Assistant** site that acts as a "Chief of Staff " that runs in cloud.

The user should be able to log in, connect their **Email** and **Calendar** (Google Workspace), and chat with the assistant to manage their day.

The Core Requirement: Dynamic Memory

The "Brain" of your agent (built on **LangGraph**) must be alive.

1. **From Chat:** If I tell the bot, "I hate 9 AM meetings," it should update the memory.
2. **From Data:** If the agent reads an email that says, "Project X is delayed," it should automatically extract that fact and update the memory without me telling it to.
3. **Retrieval:** Next time I ask to draft an email, it should know *Project X is delayed* and *I don't do 9 AMs*.

The Tech Stack

- **Frontend:** React.js / Next.js (UI just needs to be functional/clean).
- **Backend:** Python (FastAPI or Flask).
- **AI/Agent Framework:** LangChain / LangGraph. AWS agentcore Runtime / Memory for hosting agents (Optional)
- **Cloud (The "Real" Test):**
    - **AWS** is the target (ECS Fargate, Lambas, API Gateway, PostgreSQL-RDS or DynamoDB, Cloudfront, etc.).
    - **GCP** for Oauth, Gmail and Calendar connection.

    - *Note:* If AWS is impossible as it is paid service, you may use other providers (Render/Railway), but **AWS deployments will be ranked significantly higher.**
- **DevOps:** Terraform (Infrastructure as Code) and CI/CD (GitHub Actions).

Feature Checklist

1. Authentication & Google Integration

- User logs in via OAuth.
- **CRITICAL:** When setting up your Google Cloud Console for the OAuth app, add **harisankar@sentellent.com** as a **Test User**. This allows us to actually log in and test your app without you needing a verified Google App.

2. The Agent Workflow

- **Connect:** User connects Gmail/Calendar.
- **Query:** User asks, "What's important in my inbox?"
- **Action:** Agent reads mail -> **Updates Memory Graph** -> Generates Answer.
- **Action:** User says, "Reply to that email." Agent checks memory for user style/context -> Drafts/Sends email.

3. Infrastructure

- Dockerize the application.
- Write Terraform scripts to provision the resources.
- Setup a CI/CD pipeline: When you push to main, it should deploy automatically.

Recommended Build Path

To help you ship a high-quality submission within this tight deadline, we suggest building in these phases. Completing **Phase 1** alone makes your project submission-worthy and covers more than half the challenge.

Phase 1: The Foundation (Core Engineering)

- **The Agent:** Build a simple chat assistant using **LangGraph** / **Langchain** and **MCP** servers and basic Auth for the sit.
- **The Database:** Set up your DB (PostgreSQL/Vector store) to handle basic persistence.
- **The "Heavy Lift":** Write your **Terraform** scripts and set up the **CI/CD pipeline** for the Frontend and Backend deployments, with DB migrations.
- **Goal:** Get a "Hello World" agentic chat live on AWS. **If you reach this point, you have already crossed the hardest part of the ocean.**

Phase 2: The Integration (The "Chief" Skills)

- **Google Workspace:** Connect the Gmail and Calendar APIs via OAuth.

- **Action Tools:** Give the agent the ability to fetch emails or view events based on user prompts.

Phase 3: The Brain (Contextual Intelligence)

- **Long-Term Memory:** Implement dynamic memory so the agent "learns" from chat and mail history.
- **Pro Tip:** Explore **LangMem** or **AWS Bedrock's agentic memory features** to handle this efficiently.
- **Goal:** The agent should now know who the user is, their style, and their priorities without being reminded.

How to Submit

Since this is a complex project, we don't expect 100% completion. **If you can't finish everything, submit whatever you have built by the deadline.** So keeping this in mind build features step by step from easier tasks – Complete and move to challenging ones.

Fill out the attached Google Form with the following:

1. **GitHub Repo Link:** Must contain Frontend, Backend code.
2. **Live Application URL:** Where is it hosted?
3. **Proof of Cloud:**
    - Upload the screenshots of your **AWS Console** (or hosting dashboard) showing the running services.
    - Upload the screenshots of your **CI/CD pipeline** passing.

Submission Link: forms.gle/VVd9r4kemkWoiQVQ8

The Bonus

The deadline is **January 24th, 11:59 PM**. However... Sentellent is a startup. Speed matters. If you submit a solid project **2 days before the deadline**, you are indirectly telling us you ship fast. That puts you at the top of the pile.

Next Steps

Reviewing your code and deployment will take time. Shortlisted candidates who survive this round will be invited for a **technical interview** to discuss their implementation.

Good luck. Let's see what you can ship.