Assured Sentinel: Project Plan

Status: PHASE 1: BUILD

Architecture: Split Conformal Prediction (SCP) for Multi-Agent Systems

Domain: Code Security & Generation

1. Executive Summary

Assured Sentinel is a Multi-Agent System designed to solve the stochastic nature of LLM code generation. It utilizes a two-agent pattern to ensure mathematical safety guarantees.

- **The Analyst:** A high-temperature LLM agent generating Python code.
- **The Commander:** A deterministic guardrail using Conformal Prediction to reject outputs that do not meet a strict non-conformity threshold (α).

2. Technical Architecture

The Stack (Microsoft Ecosystem)

- **Development:** GitHub Codespaces (Cloud Compute) via local VS Code.
- **AI Engine:** Azure OpenAI Service (gpt-4o).
- **Orchestration:** Microsoft Agent Framework / Semantic Kernel (Python).
- **Mathematics:** mapie (Conformal Prediction), scikit-learn.
- **Judgement Logic:** bandit (Security Linter) for Non-Conformity Scoring.
- **Observability:** streamlit for live calibration dashboards.

The Data Flow

1. **User** sends query ("Write a function to parse XML").
2. **Analyst** generates candidate code.
3. **Commander** intercepts response:
   - Runs bandit analysis on code.
   - Calculates Non-Conformity Score (Inverse Security Score).
   - Compares against calibrated threshold $\hat{q}$.
4. **Decision:**
   - **Pass:** Code returned to User.

    - **Reject:** Code returned to Analyst for correction (The Loop).

---

3. Build Week Roadmap

✅ Day 0: Infrastructure & Security (Sunday)

- [x] Provision Azure OpenAI Resource.
- [x] Configure GitHub Codespaces (Dev Container).
- [x] Install dependencies (mapie, bandit, semantic-kernel, streamlit).
- [x] Secure API Keys via GitHub Secrets (No local .env files).
- [x] Verify Connectivity (SYSTEM READY check).

📅 Day 1: The Analyst (Monday)

Objective: Establish the generative engine.

- [ ] Implement analyst.py.
- [ ] Connect to Azure OpenAI using DefaultAzureCredential or env vars.
- [ ] Create the System Prompt (Persona: Senior Python Engineer).
- [ ] **Deliverable:** A script that takes a prompt and returns valid Python code.

📅 Day 2: The Scorer (Tuesday)

Objective: Define "Weirdness" (Non-Conformity Measure).

- [ ] Implement scorer.py.
- [ ] Write function to parse LLM output (extract code block).
- [ ] Run bandit programmatically on the string.
- [ ] **Deliverable:** A function score(code_str) -> float (0.0 = Secure, 1.0 = Vulnerable).

📅 Day 3: The Calibration (Wednesday)

Objective: Calculate the Safety Threshold ($\hat{q}$).

- [ ] Load Ground Truth Dataset (HumanEval or MBPP via Hugging Face).
- [ ] Run The Scorer on 100-200 calibration examples.

- [ ] Use mapie (or manual quantile calculation) to determine $\hat{q}$ for $\alpha = 0.1$ (90% confidence).
- [ ] **Deliverable:** A persisted calibration.pkl or hardcoded threshold value.

📅 Day 4: The Commander (Thursday)

Objective: Build the Logic Gate.

- [ ] Implement commander.py.
- [ ] Wrap the Analyst call in a CP verification step.
- [ ] Implement the Rejection Logic ("Sorry, this code is not assured.").
- [ ] **Deliverable:** An End-to-End script that refuses to output insecure code.

📅 Day 5: The Loop & Dashboard (Friday)

Objective: Experience & Correction.

- [ ] Implement the Correction Loop (If rejected, pass error back to Analyst).
- [ ] Build dashboard.py using Streamlit.
- [ ] Visualize the calibration histogram and live checking.
- [ ] **Deliverable:** Final Demo.

---

4. Rules of Engagement

1. **MVP First:** If it doesn't unblock the Analyst/Commander loop, it gets backlogged.
2. **Cloud Only:** No local execution of models. Respect the Surface Pro's limits.
3. **Security:** No API keys in code. Use os.getenv() only.
4. **Observability:** If we can't measure it, we can't trust it.

---

5. Context Restoration (For AI Assistants)

If starting a new session with this repo, read this section to orient yourself.

- **User Role:** AI Engineer / TPM.
- **Assistant Role:** Assured Sentinel Architect.

- **Current Phase:** Consult the checkboxes in Section 3.
- **Constraint:** We are strictly following the Split Conformal Prediction methodology.