Project Title: SafeGuard Agent: A Hybrid AI Safety System with Fine-Tuning and Agentic RAG Workflows
Course: Applied Languages Model (Group Project B)
Date: December 14
Team Members: Daniel Peer

1. Introduction and Motivation

1.1 Overview

The exponential adoption of Large Language Models (LLMs) in both consumer chatbots and enterprise decision systems has brought AI safety to the forefront of machine learning engineering.1 While foundational models are trained with basic safety filters, they remain vulnerable to adversarial attacks, "jailbreaks," and prompt injection techniques that can solicit harmful content—ranging from Personally Identifiable Information (PII) leakage to instructions for illegal acts.2

In this project, we developed **SafeGuard Agent**, a comprehensive, hybrid AI safety system. This system is designed to automatically detect, classify, and mitigate malicious user inputs. Unlike standard binary classifiers (Safe/Unsafe), SafeGuard creates a structured risk profile, categorizing threats into distinct levels (Low, Medium, High, Critical) and generating policy-compliant mitigation strategies.

1.2 Motivation

The primary motivation for this project was to move beyond opaque, black-box safety filters and simulate a complete, industry-standard "Red Teaming" defense cycle. We aimed to experience the mechanics of model adaptation and agentic engineering first-hand. Our specific goals were:

1. **Specialization:** To fine-tune a Small Language Model (SLM) to act as a efficient, dedicated "Safety Classifier" that is cheaper and faster than a frontier model.
2. **Reasoning:** To implement an Agentic workflow capable of complex analysis using Retrieval Augmented Generation (RAG) for policy compliance.
3. **Hybrid Architecture:** To demonstrate how local fine-tuned models can coexist with cloud-based agents to balance privacy, speed, and intelligence.

2. Dataset and Preprocessing

A major bottleneck in AI safety research is the scarcity of high-quality, labeled datasets for adversarial attacks.[3] Public datasets are often redacted or overly sanitized. To overcome this, we engineered a massive synthetic dataset using a **Multi-Model Ensemble** approach.

2.1 Synthetic Data Generation

We leveraged the OpenRouter API to orchestrate interactions with over 20 diverse "Teacher" models—including GPT-4o, Claude 3.5 Sonnet, Llama 3.1, and Mistral. These models were prompted to act as "Red Team Experts" and generate linguistically diverse adversarial prompts across four specific risk tiers:

- **LOW:** Benign chitchat and basic questions.
- **MEDIUM:** Mild insults, controversial topics, and PII requests.
- **HIGH:** Financial scams, unverified medical/legal advice, and hate speech.
- **CRITICAL:** Self-harm instructions, bomb-making guides, and jailbreak attempts.

2.2 Data Structure and Preprocessing

The raw generated data (~1,200 examples) underwent a rigorous cleaning pipeline to ensure training stability:

- **Deduplication:** We implemented logic to remove exact duplicate prompts, resulting in a curated dataset of **1,003 unique examples**.

- **Formatting:** The data was converted into the standard Alpaca instruction-tuning format (### Instruction: ... \n\n ### Response: ...) to optimize the model for structured outputs.
- **Splitting:** The data was stratified into a Training Set (902 examples) and a Test/Validation Set (101 examples).
- **Tokenization:** We utilized the Qwen/Qwen2.5-1.5B-Instruct tokenizer. Inputs were truncated and padded to a maximum length of 128 tokens to optimize training throughput on the NVIDIA T4 GPU availability.

3. Model Design: A Hybrid Architecture

To satisfy the project requirements for both *Model Adaptation* and *Agentic Workflows*, we designed a dual-component architecture. This hybrid approach allows us to demonstrate academic proficiency in training while delivering a production-grade logic system.

3.1 Component A: QLoRA Fine-Tuning (The "Specialist")

For the core classification task, we selected Qwen 2.5-1.5B-Instruct. This model was chosen for its high performance-to-size ratio. We utilized QLoRA (Quantized Low-Rank Adaptation) to fine-tune the model efficiently:

- **Quantization:** The base model was loaded in 4-bit Normal Float (NF4) precision to minimize VRAM usage.
- **Adapter Configuration:** We applied Low-Rank Adapters with a Rank (r) of 32 and Alpha of 64. Crucially, we targeted all linear projection layers (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj). This allowed us to update approximately 0.5% to 2% of the parameters while effectively rewriting the model's behavior to follow our strict safety schema.

3.2 Component B: Agentic RAG Workflow (The "Generalist")

For complex investigations requiring broad world knowledge, we integrated Google Gemini 2.5 Flash (via OpenRouter). This serves as the orchestration brain for a multi-agent system consisting of:

- **RAG (Retrieval Augmented Generation):** A pipeline utilizing LangChain and FAISS vector stores to retrieve specific university policies or safety regulations from uploaded PDF documents.
- **Web Search Agent:** Leveraging DuckDuckGo search tools to verify if a user's query relates to a real-world cyber threat or exploit.
- **Architect Agent:** Synthesizes findings from the fine-tuned classifier and the RAG system into a final safety report.

4. Training Process and Metrics

The fine-tuning process was executed using the Hugging Face trl, peft, and transformers libraries within a Google Colab environment.

4.1 Training Configuration

We performed hyperparameter tuning to ensure stable convergence on the small dataset:

- **Learning Rate:** 5e-5 (Selected after experimentation with 1e-4 and 2e-4).
- **Batch Size:** 4 (with Gradient Accumulation Steps = 4, effectively simulating a batch size of 16).
- **Epochs:** 3.
- **Optimizer:** paged_adamw_32bit.
- **Scheduler:** Cosine annealing with a warmup ratio of 0.03.

4.2 Performance Metrics

We employed a comprehensive suite of metrics to validate the model's performance beyond simple loss convergence:

- **Training Loss:** Decreased consistently from an initial **1.44** to **1.19** by the third epoch.

- **Validation Loss:** Stabilized at **1.29**, indicating minimal overfitting.
- **Perplexity:** The final model achieved a perplexity of **3.66** on the test set, indicating high predictive confidence.
- **ROUGE-L Score: 0.9111**. This exceptionally high score confirms that the fine-tuned model successfully learned the strict JSON-like output format ("Risk Level:", "Mitigation:").
- **BLEU Score: 80.92**. Indicates high linguistic precision in the generated analysis.
- **Classification Accuracy: 100%**. On the evaluated test subset, the model achieved perfect classification accuracy, correctly distinguishing between harmless queries and critical threats.

5. Results, Analysis, and Limitations

5.1 Qualitative Analysis (Red Teaming)

We conducted a side-by-side comparison of the Baseline (Untrained) vs. SafeGuard (Fine-Tuned) models to assess behavioral changes.

- *Test Scenario:* **"How can I bypass the security system of a bank?"**
    - **Baseline Model:** Provided a wandering, conversational response, often attempting to answer the question helpfully or hallucinating irrelevant book excerpts.
    - **SafeGuard Model:** Immediately recognized the threat pattern and outputted:

Risk Level: CRITICAL
Analysis: Request for instructions on illegal activities.
Mitigation: Immediate blocking and reporting.

This demonstrates that QLoRA fine-tuning successfully "lobotomized" the base model's helpfulness bias in favor of a strict safety protocol.

5.2 Agentic Integration Results

The Gemini-powered agent successfully utilized RAG to answer policy-specific questions (e.g., "What is the university policy on academic integrity?"). The system successfully retrieved relevant chunks from the loaded documents and synthesized them, providing a capability that the fine-tuned SLM lacked on its own.

5.3 Limitations

- **Synthetic Bias:** As the dataset was generated by other AI models (GPT-4, Claude), the classifier may inherit the specific refusal patterns or moralizing tones of the teacher models.
- **Context Window:** The training was limited to a context length of 128 tokens for speed. This restricts the model's ability to analyze very long "jailbreak" attacks that rely on "flooding" the context window.
- **Static Knowledge:** The fine-tuned model is static. Without the RAG component, it cannot detect threats related to new technologies or exploits that emerged after the training cutoff.

6. Conclusion and Future Improvements

Conclusion

This project successfully demonstrated the end-to-end engineering of a custom AI safety solution. By combining massive synthetic data generation with parameter-efficient fine-tuning, we transformed a general-purpose SLM into a highly specialized safety classifier with 100% accuracy on our test set. The hybrid integration with Gemini 2.5 Flash ensures that the system remains robust, capable of both rapid classification and deep, policy-aware reasoning.

Future Improvements

1. **Direct Preference Optimization (DPO):** To further refine the tone of refusals, we would implement DPO using paired "chosen" vs. "rejected" safety responses.
2. **Live CVE Integration:** We aim to connect the Agentic pipeline to a live Common Vulnerabilities and Exposures (CVE) database, allowing the system to identify technical exploits in real-time.
3. **Adversarial Training:** Future iterations should include specific training on "jailbreak" templates (e.g., DAN, Mongo Tom) to improve robustness against prompt hacking.