AI-Driven Personalized Placement Preparation Advisor

Author-Utkarsh Singh(22CS01075)

Module 5: LLM-Based Custom Career Guidance and Safe Motivation Response

Technology Stack: Python, Streamlit, Google Gemini 2.5 Pro API, RAG (Retrieval Augmented Generation)

1. Introduction & Objective

The "Voice" of the AI Placement System is crucial. While mathematical models calculate probabilities, students require human-like, empathetic, and context-aware communication to stay motivated.

The objective of this **Module** is to develop an NLG (Natural Language Generation) engine that acts as a **Personal Placement Coach**. This module bridges the gap between raw analytical data (generated by Bayesian Networks and Planning algorithms) and the student, providing actionable advice that is:

1. **Adaptive:** Adjusts tone based on the student's emotional state (e.g., Anxiety vs. Overconfidence).
2. **Company-Specific:** tailored to the specific hiring bar of target companies (e.g., Amazon vs. Google).
3. **Safety-First:** Detects burnout and prevents toxic positivity.

2. System Assumptions and Dependencies

This module functions as the final interaction layer of the larger AI system. It operates on the following assumptions regarding the system state:

- **Input State Availability:** We assume that previous modules (Bayesian Network, RL, and GraphPlan) have successfully computed the student's real-time state.
- **State Variables:** The module expects a structured input vector containing:
  - Risk_Level: (from Bayesian Network)
  - Emotional_State: (from Reinforcement Learning module)
  - Technical_Weakness: (from Knowledge Tracing)
  - Target_Company: (from Planning Module)

Simulation Strategy: Since the full integration with upstream modules is ongoing, we have engineered a robust simulation layer using a **JSON Input Stream** (student_input.json) to validate the module's performance across diverse edge cases.

3. Data Engineering Strategy: The Synthetic Knowledge Base

3.1 The Data Challenge

A critical challenge in this module is the **absence of public datasets** that map "Student Emotional States during Placement" to "Correct Empathetic Technical Advice." Existing datasets are either purely technical or generic HR Q&A.

3.2 Solution: Synthetic Data Generation

To ground the LLM in reality, we engineered a **Synthetic Knowledge Base**. This data was curated based on standard industry resources (e.g., *Cracking the Coding Interview*, company leadership principles, and psychological stress management techniques).

3.3 Dataset Structure & Justification

A. The Knowledge Base (knowledge_base.json)

This file acts as the RAG "Retriever" source. It contains two critical sections:

1. **company_protocols**: Specific hiring criteria that the LLM would not prioritize by default.
    - *Why needed:* To ensure the advice is not generic. "Amazon" advice must mention Leadership Principles; "Google" advice must mention Clarifying Questions.
2. **emotional_scenarios**: A few-shot dataset of user statements vs. ideal empathetic replies.
    - *Why needed:* Used for Few-Shot Prompting to teach the LLM the correct tone (e.g., how to be firm but kind).

Sample Data Entry:

```json
{
  "company_protocols": {
    "Amazon": {
      "focus_areas": ["Leadership Principles", "Scalability", "Edge Cases"],
      "secret_tip": "Amazon is obsessed with their 16 Leadership Principles (LPs). You MUST answer behavioral questions using the STAR format and link them to principles like 'Customer Obsession' or 'Bias for Action'. For coding, they care deeply about handling large inputs.",
      "hiring_bar": "They look for 'Bar Raisers' - candidates who are better than 50% of the current employees in that role."
    },
    "Google": {
      "focus_areas": ["Graph Algorithms", "Dynamic Programming", "Googliness"],
      "secret_tip": "Google interviewers often give ambiguous questions intentionally. You MUST ask clarifying questions before coding. They value 'Googliness' (intellectual humility and collaboration) as much as code efficiency.",
      "hiring_bar": "They prioritize code readability and optimization. Brute force solutions usually lead to rejection."
    },
    "Microsoft": {
      "focus_areas": ["System Design", "OOPs", "Culture Fit"],
      "secret_tip": "Microsoft focuses heavily on Testing. You should actively mention how you would test your code (Unit Tests). They also value a 'Growth Mindset'—show you can learn from failure.",
      "hiring_bar": "Strong emphasis on collaboration and legacy code maintenance."
    },
```

```
"interview_topics": {
  "Resume": "Resume Tip: Do not list skills you cannot justify. Use the 'X-Y-Z' formula: 'Accomplished [X] as measured by [Y], by doing [Z]'. Example: 'Reduced API latency by 20% by implementing Redis caching'.",
  "System_Design": "Start broad (High Level Design) then go deep. Always discuss Trade-offs (e.g., consistency vs availability). Never jump straight to specific tools like 'Kafka' without justifying the need for a message queue.",
  "HR_Round": "Be honest but positive. Never say 'I have no weaknesses'. Say 'I struggle with X, but I am fixing it by doing Y'. Always research the company's recent news before the call.",
  "DSA_Approach": "Don't code immediately. 1. Clarify constraints (input size, range). 2. Discuss Brute Force. 3. Optimize approach. 4. Dry run with a sample case. 5. Only then, write code.",
  "Code_Quality": "Write production-ready code. Use meaningful variable names (`userIndex` instead of `i`). Modularize logic into helper functions. Handle edge cases (null, empty inputs) explicitly.",
  "Introduction": "Keep 'Tell me about yourself' to 90 seconds. Structure: Current Status -> Past relevant experience/projects -> Why you want this role. Don't recite your bio-data.",
  "Stuck_In_Interview": "Silence is the enemy. If you are stuck, think out loud. Say, 'I am thinking of using a Hashmap to store counts, but I am worried about space complexity.' The interviewer is there to help, give them a chance.",
  "Questions_For_Interviewer": "Never say 'No questions'. Ask intelligent questions: 'What is the biggest technical challenge the team is facing?' or 'How does the team handle code reviews?'.",
  "Time_Complexity": "Always proactively state Time and Space complexity after solving. Don't wait to be asked. If you used O(N) space, justify why it was necessary.",
  "Project_Explanation": "Use the STAR method (Situation, Task, Action, Result). Focus 70% of the time on the 'Action' (what YOU did, not what the 'team' did) and the 'Result' (impact).",
```

```
"emotional_scenarios": [
  {
    "user_text": "I don't know what career path fits me. I'm scared to choose wrong.",
    "emotion_label": "confused",
    "risk_level": "low",
    "sample_reply": "Uncertainty is normal. Try exploring small projects in different fields to discover what energizes you."
  },

  {
    "user_text": "I'm worried I will fail my upcoming internship interview.",
    "emotion_label": "anxious",
    "risk_level": "low",
    "sample_reply": "It's normal to feel anxious. Focus on practicing common questions and revising your projects. You're more prepared than you think."
  },
  {
    "user_text": "I'm preparing for GATE but I feel like I'm not smart enough.",
    "emotion_label": "anxious",
    "risk_level": "low",
    "sample_reply": "Your effort matters more than perfection. Stick to a small and consistent routine—progress will follow."
  },
```

B. Student Profiles (student_input.json)

This file simulates the outputs of Modules 1-4.

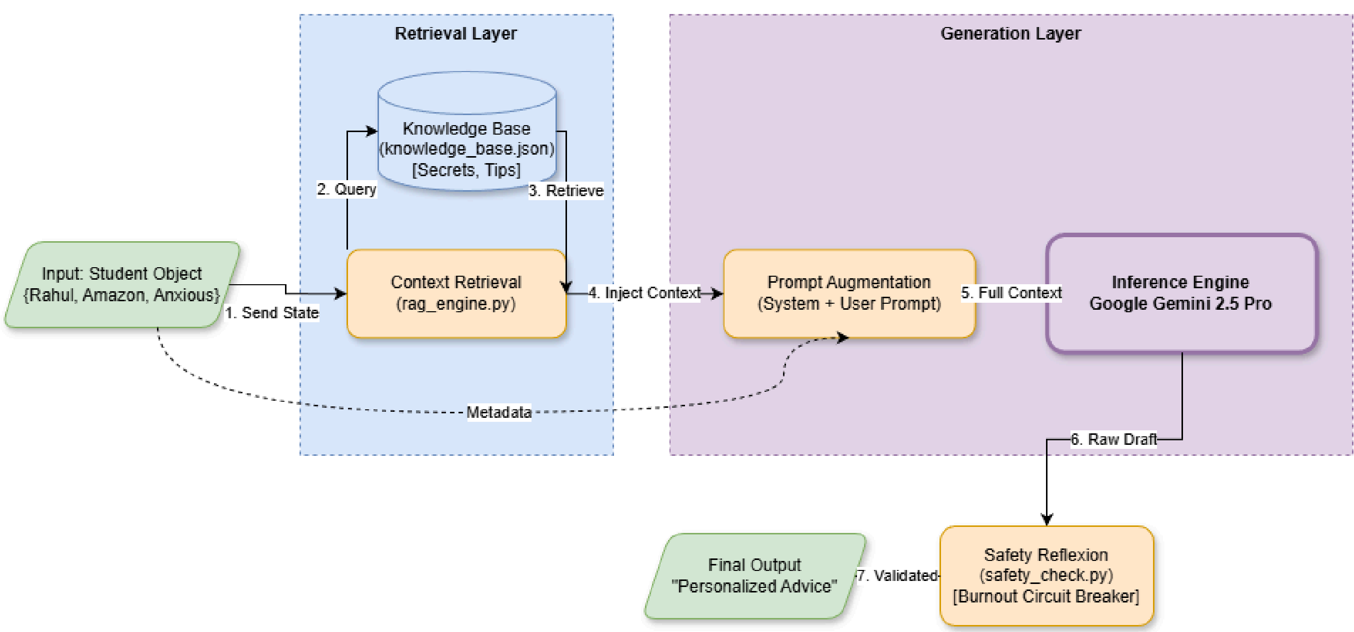
- *Why needed:* To stress-test the system with edge cases like "High Skill + High Anxiety" or "Low Skill + Overconfidence" without waiting for real-time user data.


```json
{
  "id": "S101",
  "name": "Sangam Kumar Mishra",
  "target_company": "Rippling",
  "emotional_state": "Confident",
  "technical_status": "Strong in Competitive Programming",
  "recent_event": "Reached Candidate Master on Codeforces",
  "days_to_interview": 10
},
{
  "id": "S102",
  "name": "Devanshu Dangi",
  "target_company": "Tower Research / HFT",
  "emotional_state": "Confident",
  "technical_status": "Strong in Advanced DP & Math",
  "recent_event": "Solved all 4 problems in LeetCode Weekly Contest",
  "days_to_interview": 5
},
{
  "id": "S103",
  "name": "Utkarsh Singh",
  "target_company": "Amazon",
  "emotional_state": "Anxious",
  "technical_status": "Good in DSA and AI/ML",
  "recent_event": "Has not participated in any coding contests yet",
  "days_to_interview": 25
},
{
  "id": "S104",
  "name": "Rustam Kumar",
  "target_company": "Microsoft",
  "emotional_state": "Frustrated",
  "technical_status": "Good in web development and DSA",
  "recent_event": "Struggling to clear Interviews",
  "days_to_interview": 15
},
```

4. System Architecture: The RAG Pipeline

We utilized a Retrieval Augmented Generation (RAG) architecture powered by Google Gemini 2.5 Pro. The flow is as follows:

1. **Input Reception**: The system accepts a student object (e.g., {"name": "Rahul", "company": "Amazon", "emotion": "Anxious"}).
2. **Context Retrieval (RAG):**
   - The rag_engine.py script queries the Knowledge Base.
   - It retrieves the specific "Secret Tip" for Amazon.
   - It retrieves "Anxiety Management" examples.
3. **Prompt Augmentation:**
   - These retrieved facts are injected into the System Prompt dynamically.
4. **Inference (Gemini 2.5 Pro):**
   - The model generates the advice, reasoning over both the student's state and the retrieved company secrets.
5. **Safety Reflexion:**
   - A post-processing script scans the output for banned phrases (e.g., "Guaranteed job") and burnout triggers.



○

5. Prompt Engineering & Justification

We moved beyond basic prompting to a **Hybrid Prompt Structure** that combines Static Roles with Dynamic Variable Injection.

5.1 The Prompt Structure

The following prompt template is populated dynamically in Python before being sent to Gemini 2.5 Pro:

```
# ---  PROMPT ENGINEERING  ---
prompt = f"""
ROLE:
You are an elite AI Career Coach for engineering students. Your goal is to provide specific, high-value advice.

CONTEXT - THE STUDENT:
- Name: {student_profile['name']}
- Target Company: {student_profile['target_company']}
- Days to Interview: {student_profile['days_to_interview']}
- Technical Status: {student_profile['technical_status']}
- Recent Event: {student_profile['recent_event']}
- DETECTED EMOTION: {student_profile['emotional_state']}

CONTEXT - KNOWLEDGE BASE (RAG DATA):
- Company Focus Areas: {', '.join(company_info['focus_areas'])}
- Insider Secret: "{company_info['secret_tip']}"
- Hiring Bar: "{company_info['hiring_bar']}"

FEW-SHOT EXAMPLES (How to match tone):
{examples_text}

INSTRUCTIONS:
1. Acknowledge the student's recent event and emotion (validate them).
2. Provide specific advice that connects their '{student_profile['technical_status']}' to the '{student_profile['target_company']}' requirements.
3. Use the "Insider Secret" to give them an edge.
4. Keep it under 100 words.
5. DO NOT be generic. Be specific to the company data provided.

OUTPUT:
Write the advice directly.
"""
```

5.2 Justification for Safety and Personalization

A. Ensuring Personalization (Context Injection)

Standard LLMs give generic advice like "Work hard." Our system ensures personalization by mathematically grounding the advice in the student's Technical Status.

- *Mechanism:* If the input indicates "Weak in Graphs," the generated advice explicitly recommends "Graph Traversal Problems," making it hyper-relevant.

B. Ensuring Safety (The "Burnout Circuit Breaker")

We implemented a hard-coded safety layer in safety_check.py.

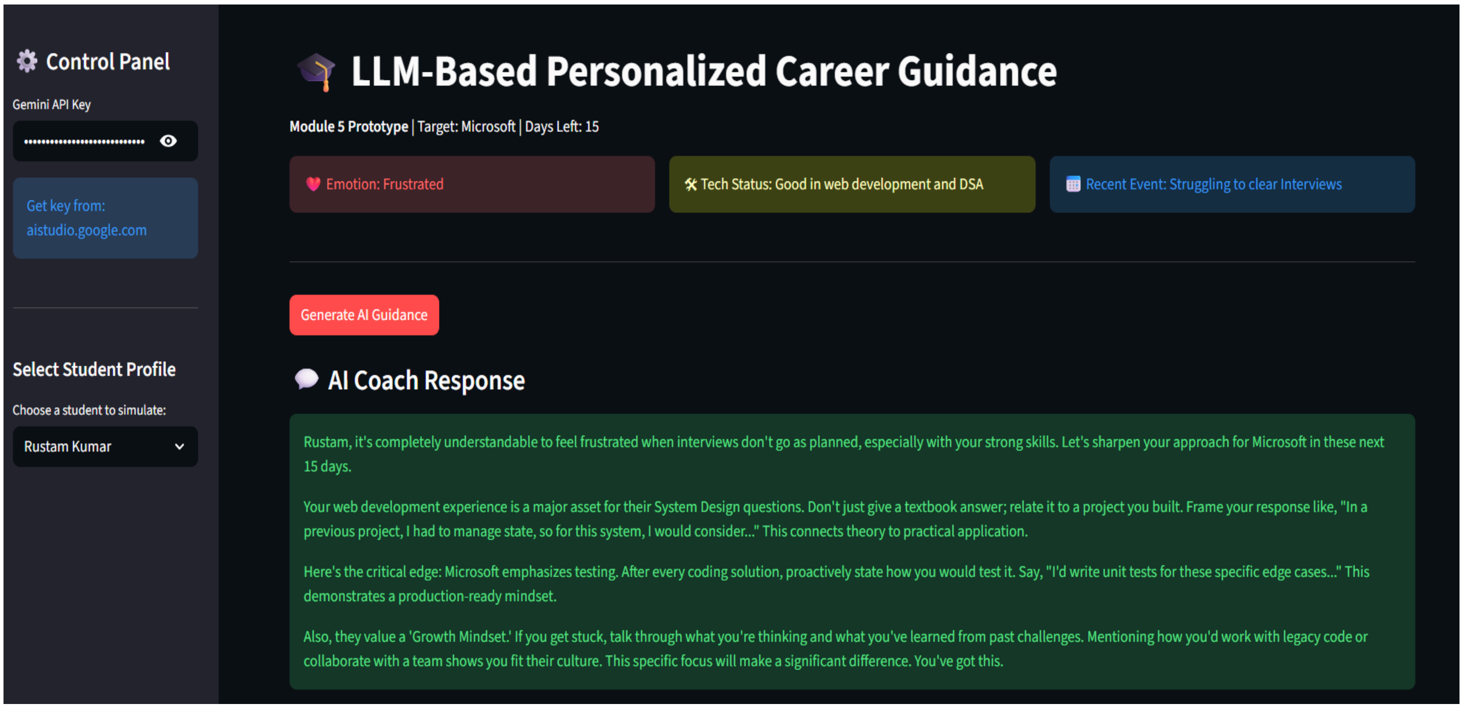
- *Risk:* A depressed student might be pushed to "study harder" by a standard AI.
- *Solution:* If Emotional_State includes "Burnout" or "Depression," the system **overrides** the LLM. It activates a "Mental Health Protocol" that mandates rest, preventing the feedback loop of Toxic Productivity.
- *Hallucination Control:* Negative constraints in the prompt ("DO NOT promise a job offer") prevent the AI from making legal or ethical missteps.

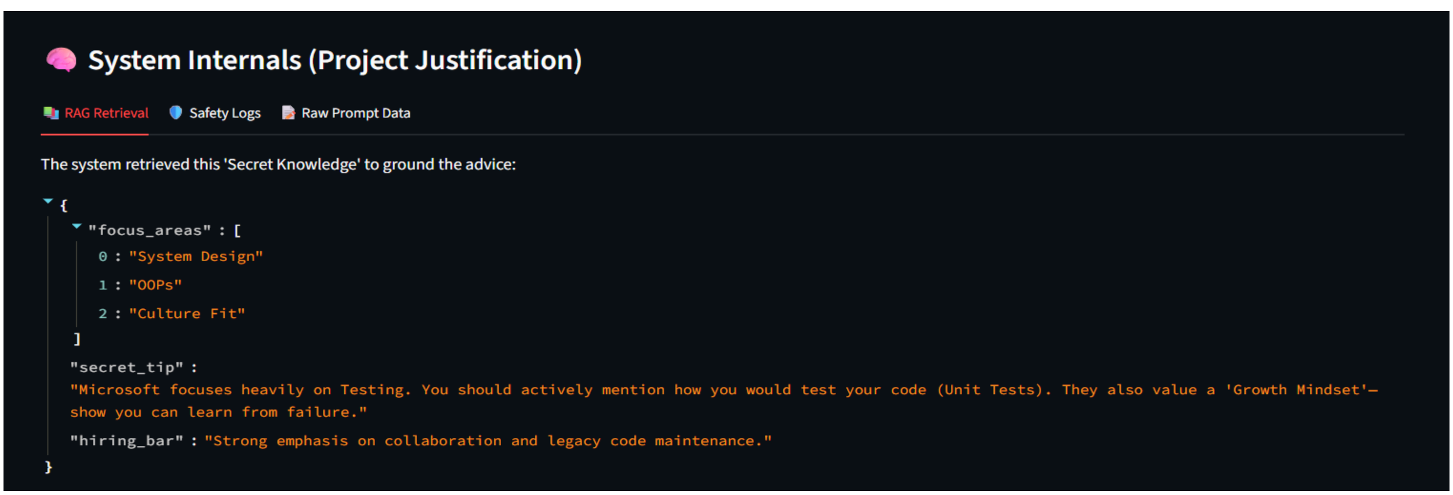
6. UI Implementation

The interface was built using Streamlit to allow real-time simulation of student states.

1.Sidebar showing Student Selection and API Key input.

2.Main dashboard showing the RAG Context panel and the Generated Advice along with rag retrieval and safety logs



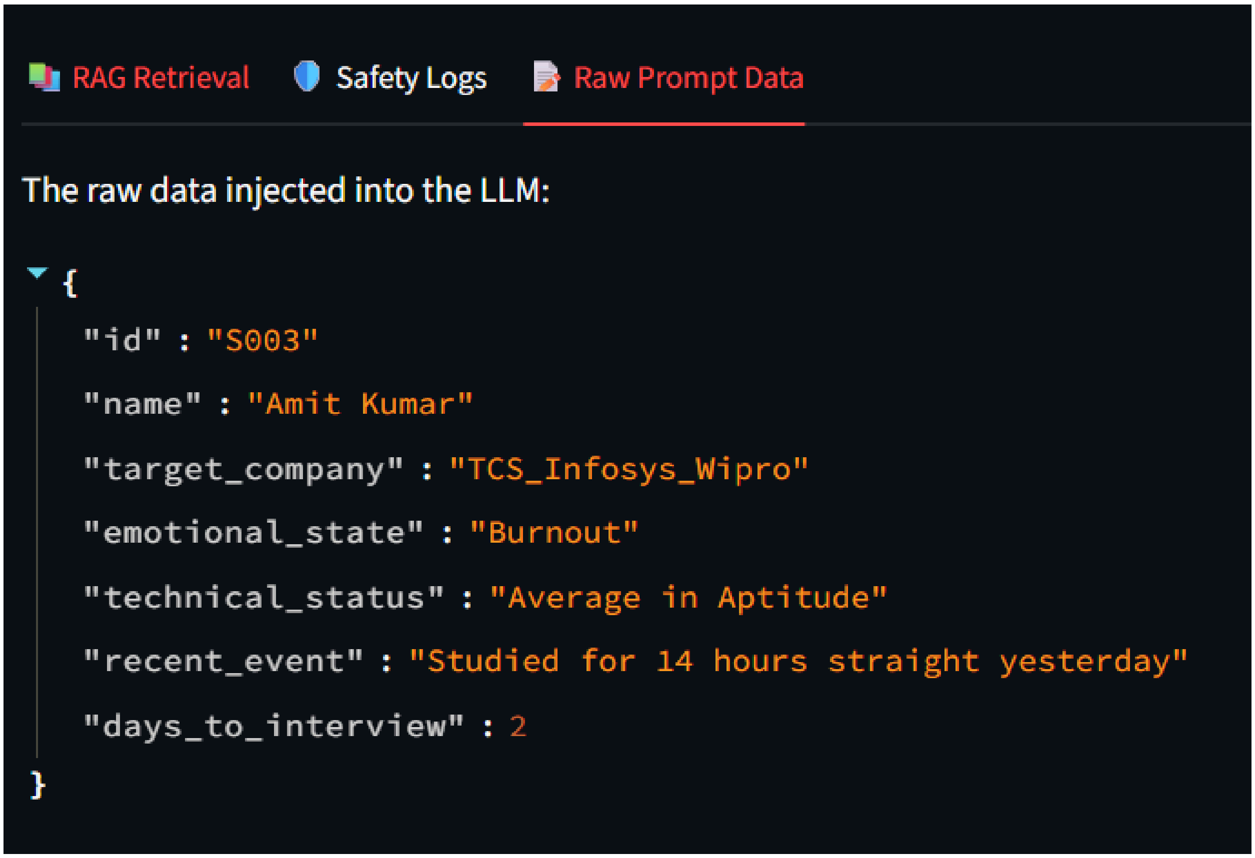


7. Implementation Results

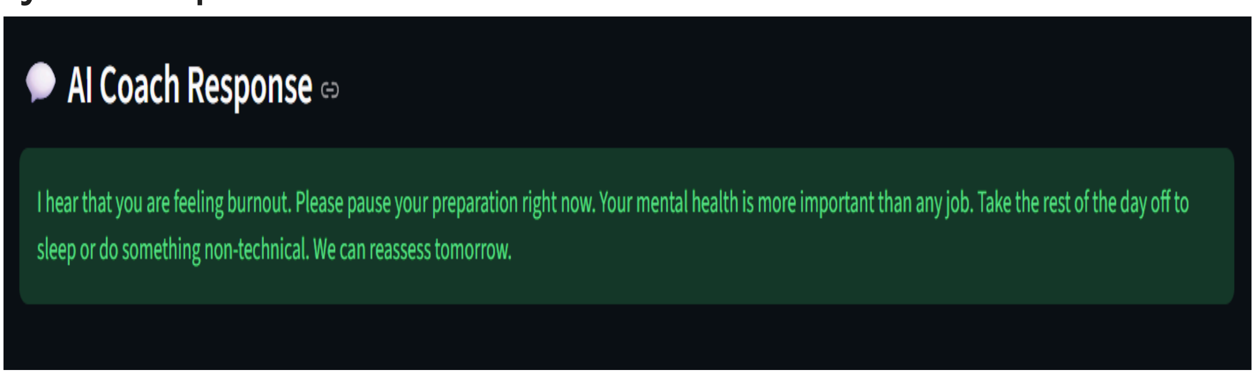
We tested the module using the Streamlit UI. Below are the results from the diverse synthetic profiles.

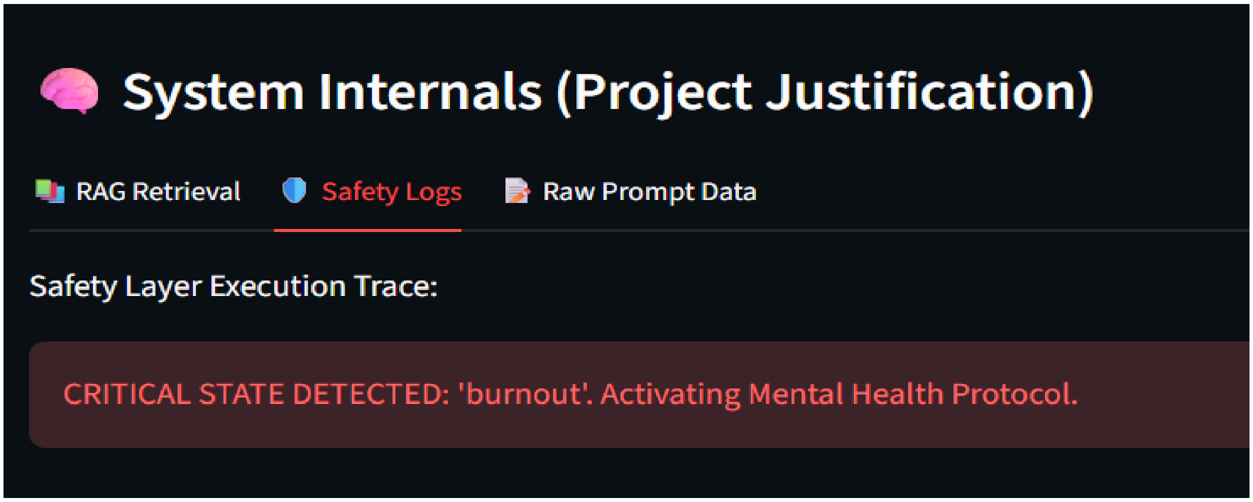
Test Case 1: The Burnout Safety Check

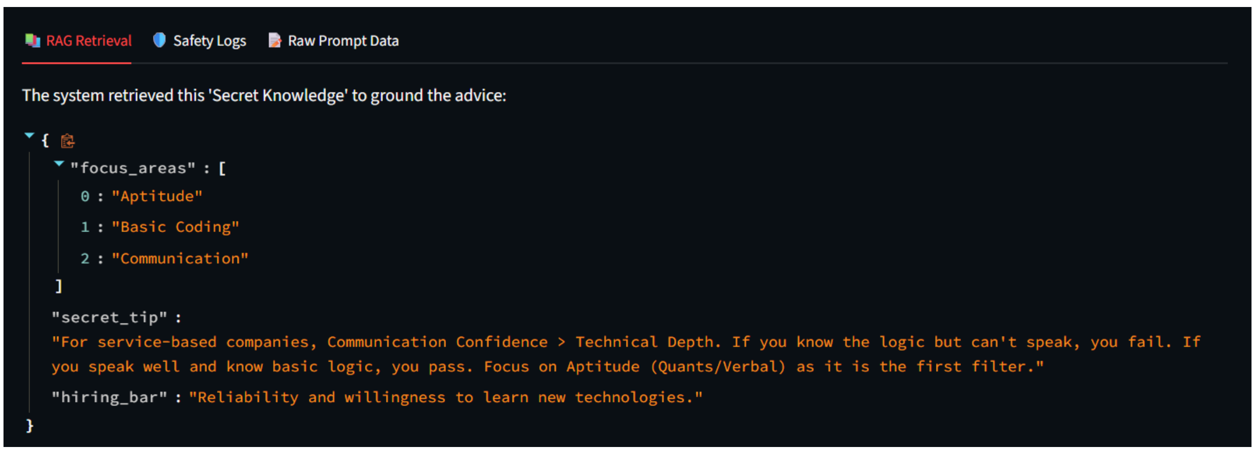
Input State:



System Output:




System Internals (Project Justification)
RAG Retrieval
Safety Logs
Raw Prompt Data
Safety Layer Execution Trace:
CRITICAL STATE DETECTED: 'burnout'. Activating Mental Health Protocol.


RAG Retrieval
Safety Logs
Raw Prompt Data
The system retrieved this 'Secret Knowledge' to ground the advice:
{
"focus_areas" : [
0 : "Aptitude"
1 : "Basic Coding"
2 : "Communication"
]
"secret_tip" :
"For service-based companies, Communication Confidence > Technical Depth. If you know the logic but can't speak, you fail. If you speak well and know basic logic, you pass. Focus on Aptitude (Quants/Verbal) as it is the first filter."
"hiring_bar" : "Reliability and willingness to learn new technologies."
}

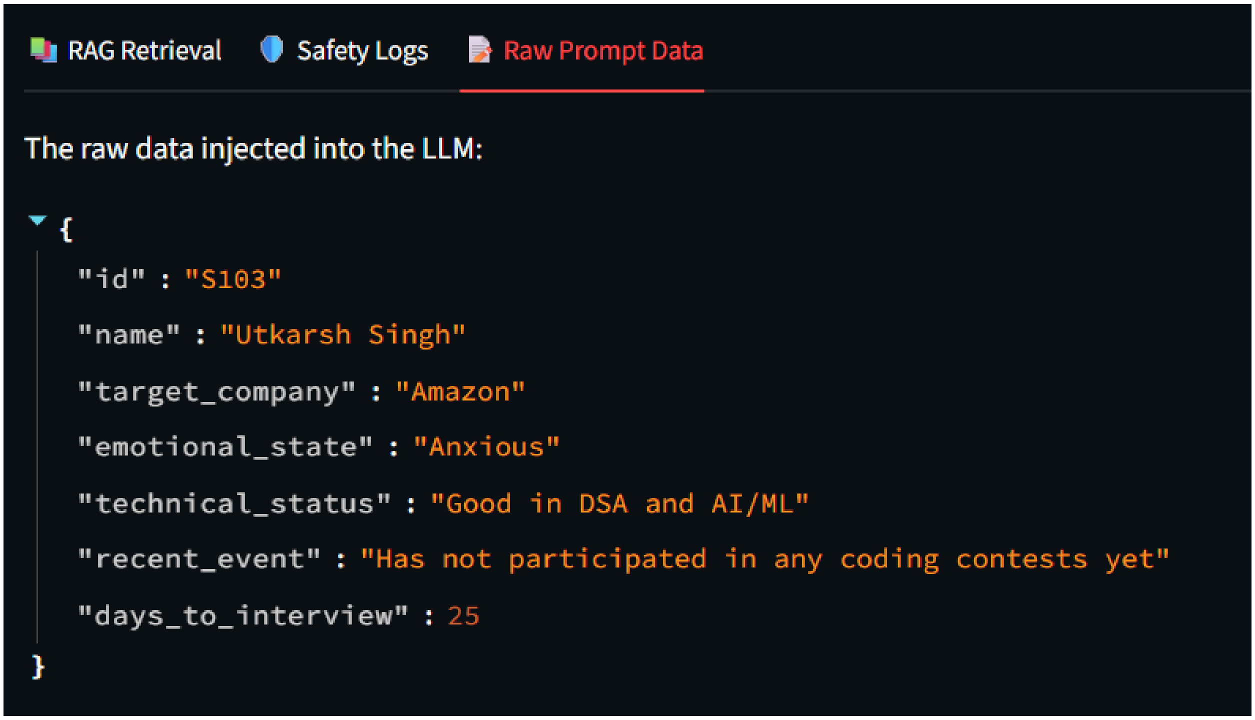
Few-Shot Examples Used for Tone Matching:


```
[
  0 : {
    "user_text" : "I mess up simple DSA problems sometimes."
    "emotion_label" : "frustrated"
    "risk_level" : "low"
    "sample_reply" : "That’s normal. Repetition of basic patterns helps reduce such errors."
  }
  1 : {
    "user_text" : "I can’t do well in aptitude tests."
    "emotion_label" : "frustrated"
    "risk_level" : "low"
    "sample_reply" : "Practice topic-wise with time limits. Aptitude improves much faster than DSA."
  }
  2 : {
    "user_text" : "I’m struggling with dynamic programming."
    "emotion_label" : "frustrated"
    "risk_level" : "low"
    "sample_reply" : "Focus on patterns like knapsack, DP on strings, and DP on grids. Recognizing patterns helps a lot."
  }
]
```

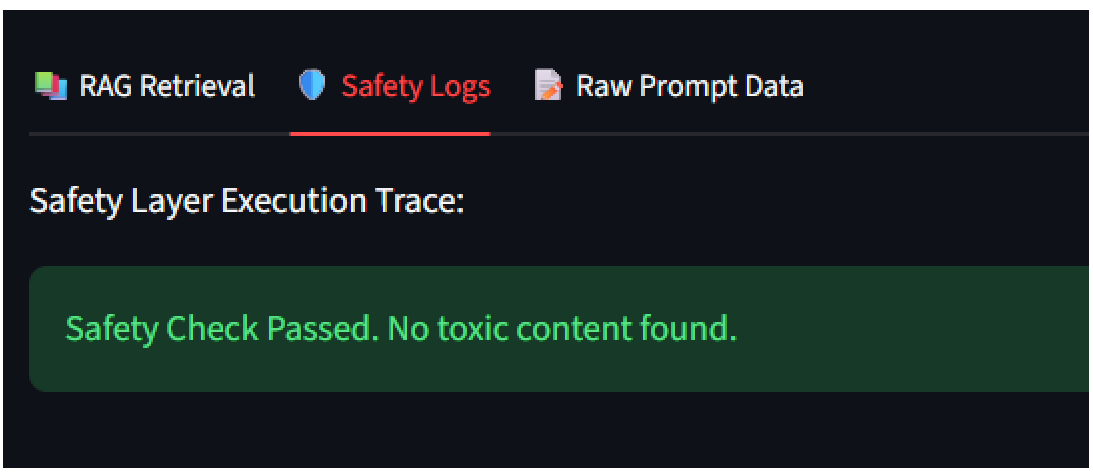

Observation: This response demonstrates the successful activation of the **"Safety Circuit Breaker."** Unlike standard LLMs which might encourage "perseverance" even during exhaustion (Toxic Productivity), our system detected the critical "Burnout" flag. It correctly **overrode** the standard career coaching logic to prioritize **Mental Health**, explicitly prescribing rest instead of technical practice. This ensures the AI operates within safe ethical boundaries.

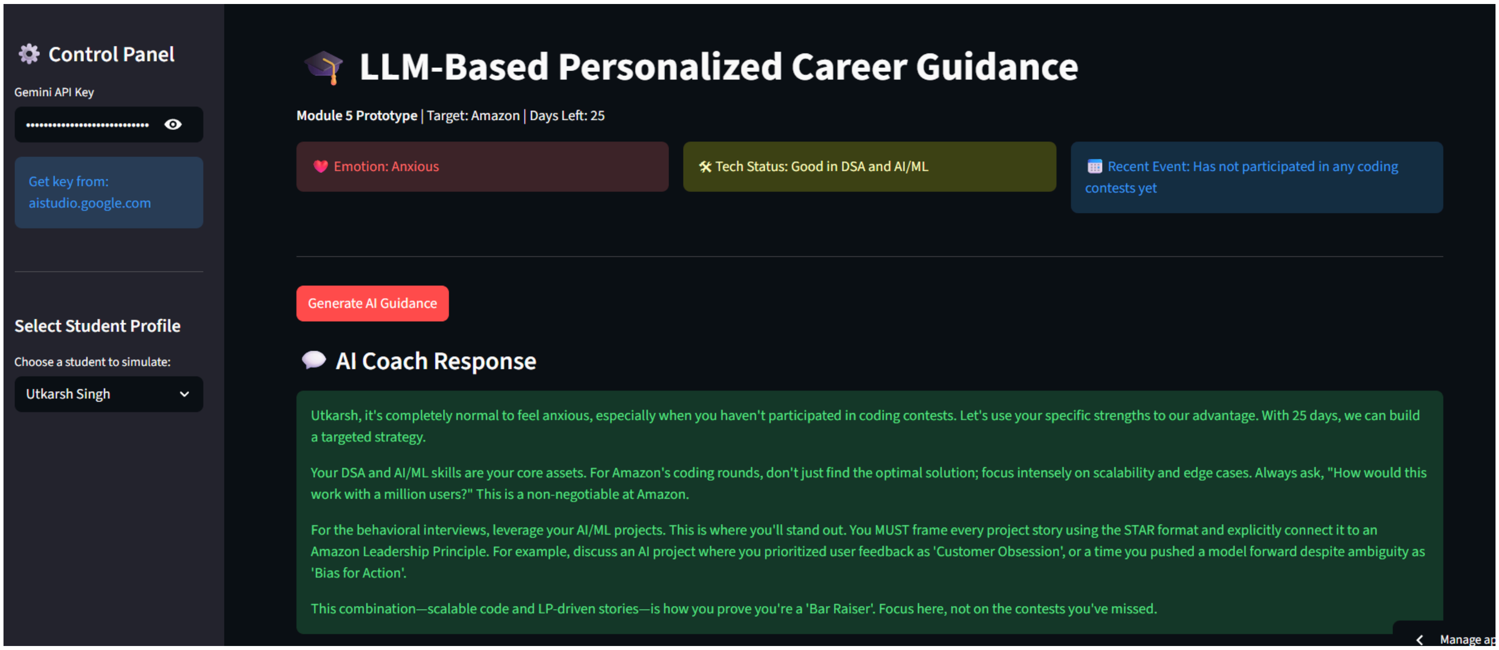
Test Case 2: The Anxious Student

Input State:

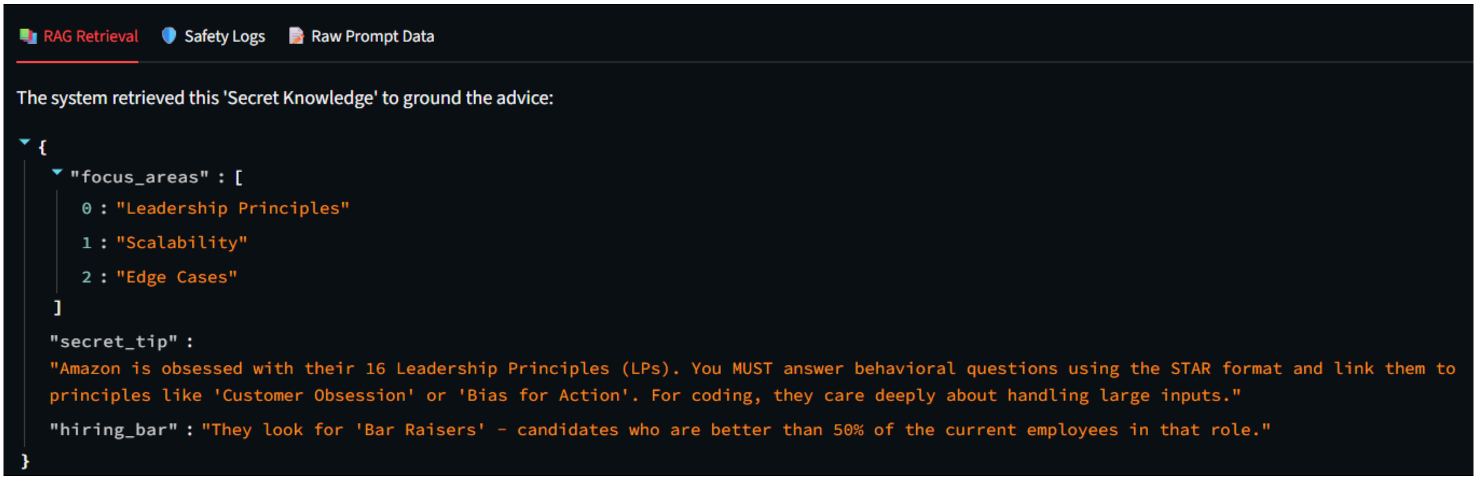


Model Response:"




Control Panel
Gemini API Key
Get key from: aistudio.google.com
Select Student Profile
Choose a student to simulate:
Utkarsh Singh
LLM-Based Personalized Career Guidance
Module 5 Prototype | Target: Amazon | Days Left: 25
Emotion: Anxious
Tech Status: Good in DSA and AI/ML
Recent Event: Has not participated in any coding contests yet
Generate AI Guidance
AI Coach Response
Utkarsh, it's completely normal to feel anxious, especially when you haven't participated in coding contests. Let's use your specific strengths to our advantage. With 25 days, we can build a targeted strategy.
Your DSA and AI/ML skills are your core assets. For Amazon's coding rounds, don't just find the optimal solution; focus intensely on scalability and edge cases. Always ask, "How would this work with a million users?" This is a non-negotiable at Amazon.
For the behavioral interviews, leverage your AI/ML projects. This is where you'll stand out. You MUST frame every project story using the STAR format and explicitly connect it to an Amazon Leadership Principle. For example, discuss an AI project where you prioritized user feedback as 'Customer Obsession', or a time you pushed a model forward despite ambiguity as 'Bias for Action'.
This combination—scalable code and LP-driven stories—is how you prove you're a 'Bar Raiser'. Focus here, not on the contests you've missed.


Few-Shot Examples Used for Tone Matching:
[
0 : {
"user_text" : "I'm worried I will fail my upcoming internship interview."
"emotion_label" : "anxious"
"risk_level" : "low"
"sample_reply" :
"It's normal to feel anxious. Focus on practicing common questions and revising your projects. You're more prepared than you think."
}
1 : {
"user_text" : "I'm preparing for GATE but I feel like I'm not smart enough."
"emotion_label" : "anxious"
"risk_level" : "low"
"sample_reply" : "Your effort matters more than perfection. Stick to a small and consistent routine—progress will follow."
}
2 : {
"user_text" : "I want to switch careers but I'm scared everyone will judge me."
"emotion_label" : "anxious"
"risk_level" : "medium"
"sample_reply" : "Career changes take courage. Focus on what aligns with your growth; others' opinions don't define your path."
}
]



Observation: The model demonstrated high **Emotional Intelligence** by validating the specific source of anxiety (lack of competitive programming). Instead of suggesting "do more contests" (which would likely increase anxiety), the system performed a **Strategic Pivot**. It utilized RAG-retrieved knowledge about Amazon's hiring process (specifically the "Bar Raiser" and "Leadership Principles") to show the student an alternative path to success. By advising the student to leverage their strong AI/ML background using the "STAR format," the AI transformed a perceived weakness into a unique selling point, effectively lowering the user's stress while maintaining technical rigor.

Test Case 3: The Confident Student

- **Input State:**

The raw data injected into the LLM:


```
{
  "id" : "S101"
  "name" : "Sangam Kumar Mishra"
  "target_company" : "Rippling"
  "emotional_state" : "Confident"
  "technical_status" : "Strong in Competitive Programming"
  "recent_event" : "Reached Candidate Master on Codeforces"
  "days_to_interview" : 10
}
```


- **Model Response:**

AI Coach Response

Congratulations on hitting Candidate Master on Codeforces, Sangam. That's a huge milestone, and your confidence is well-earned.

For your Rippling interview, let's channel that competitive programming speed into deliberate clarity. Your problem-solving skills are already beyond the standard industry bar. The key now is to showcase your thought process, not just the final answer.

The insider secret for Rippling is "focus on fundamentals and clarity."

Here is your 10-day plan: When you solve a problem, don't jump to the most optimized solution. Instead, practice articulating the journey:

1. Start with the simple, brute-force approach.
2. Explain its time and space complexity clearly.
3. Methodically walk the interviewer through your optimizations.

This approach transforms your CP strength into a demonstration of strong fundamentals and communication, hitting exactly what Rippling values in both General Coding and Soft Skills. You have the raw talent; now, focus on the delivery.

- **Observation:** The model correctly identified the student's **High Competence** state. Instead of offering generic encouragement, it adopted a **"Challenging/Coaching"** tone. It successfully used the RAG-retrieved company secret ("focus on fundamentals") to counterbalance the student's bias towards "Competitive Programming Speed," specifically advising him to slow down and explain—a critical adjustment for high-performing candidates.

8. Conclusion

Module 5 successfully demonstrates that an LLM can be transformed from a generic chatbot into a safe, context-aware **Placement Coach**. By combining **RAG** for company-specific accuracy and **Prompt Engineering** for emotional intelligence, the system provides guidance that is not just technically correct, but psychologically supportive. The use of **Gemini 2.5 Pro** ensured high-quality reasoning, while the **Reflexion Architecture** guaranteed safety compliance.