Given Problem statement :

"LLM-powered Debate Moderator"
Premise:
Simulate a 3-agent debate between personas (e.g., engineer, designer, PM) on a product
decision (e.g., "Should we use microservices?").
Your job: build the LLM-based moderator that:
● Keeps conversation on track
● Resolves misunderstandings
● Summarizes outcomes clearly
Deliverables:
1. Design agent structure (planning or turn-based)
2. Logic for summarization, conflict resolution
3. Example transcripts showing conversation handling
4. Your own critique: where does this break?

Defining Scope of Project

The scope of this project is defined with reference to the Lokal App. The current operational state of the Lokal App is considered as contextual input to ensure that all participating agents are informed while making product-level decisions. This approach enables the generation of more focused, context-aware discussions and outcomes throughout the simulated debate.

The Agents Considered

Product Manager
Role and Responsibilities:

- Defines the strategic direction and feature roadmap to address fake AI uploads.
- Aligns business goals (like trust, retention, growth) with user needs and technical feasibility.
- Prioritizes trade-offs between speed of execution, moderation accuracy, and user experience.
- Drives the team toward decisions that scale while ensuring product-market fit.

Designer (UI/UX)
Role and Responsibilities:

- Crafts user interfaces that convey rejection reasons clearly, respectfully, and emotionally intelligently.
- Ensures that moderation feedback, notification flows, and appeals are intuitive and non-alienating.
- Advocates for the emotional impact on users, especially during false positives.
- Proposes UX elements (e.g., animations, icons, microcopy) to guide user behavior without friction.

Engineer (Software / AI-ML)
Role and Responsibilities:

- Evaluates the feasibility and efficiency of implementing moderation logic, rejection pipelines, and scalable feedback systems.
- Handles integration of AI/ML models for detecting fake uploads, including edge cases and performance optimization.
- Proposes fallback systems for devices with limitations and ensures low latency across experiences.
- Flags technical debt and scalability risks in any major redesigns.

Moderator (LLM Agent)
Role and Responsibilities:

- Moderates the debate, ensuring structured and turn-based contributions from each agent.
- Keeps the conversation on track and encourages agents to build upon—not repeat—previous points.
- Identifies and resolves conflicts (e.g., between UX vision and engineering feasibility).

- Summarizes and evaluates outcomes after each round and determines when the debate should end.

- Provides a final strategic recommendation based on the conversation history.

Defining 'Product Decision' Scope :

- Current state of the product (Lokal App), features, User Flow given as input,U se cases are considered and given as Input
- Identifying a possible problem from Product Research and trying to Generate a Conversation based on it
- Current state of the product
- Considering a Real -Problem faced in Lokal App on which agents are trying to make product decision's low-level and high-level product decisions
- This is done to Get a very high quality debate

The Problem Considerd :
Spread of Fake news by user's (Users Uploading fake content) in Lokal App, The agents know the Current State of product
Now Agents will Dicuss based on the Test Cases(Prompts) Given:

Debate Structure & Rules

Participants
PM (Product Manager) – focuses on strategy and prioritization.

Designer – focuses on UX, visuals, and user flow.

Engineer – focuses on technical feasibility and implementation.

Moderator (LLM) – oversees the flow, resolves conflicts, decides when to stop, and summarizes.

How Each Debate Round Works

Topic Selection:

Each round covers a sub-topic (e.g., notifications, feedback, tech feasibility).
The topic changes every round by cycling through a predefined list.

Turn-Based Interaction:
Each role speaks in this order: PM → Designer → Engineer.

Each role responds based on:
The current topic.
The last 3 turns of conversation.
A strict instruction to provide new, unique, concise insights (1–2 sentences only).
Redundant responses are retried up to 3 times.

Enforcement:
All previous sentences are tracked and
Repeated or vague sentences are flagged and rejected before proceeding.

Moderator Responsibilities
Flow Control:
After every round, the moderator reviews recent dialogue.
It decides if the discussion is still valuable or if the debate should stop and resolves conflicts if required
It can stop anytime before the hard cap of 20 rounds.

Final Summary & Decision:
Once the debate ends, the moderator generates a 5-sentence summary.
It highlights key points from each stakeholder and gives a final product recommendation.

What is Delivered :
Realistic simulation of multi-disciplinary product discussions.
Enforces brevity, clarity, and diversity of thought.
Ensures discussion ends meaningfully, not arbitrarily.
Allows easy RAG integration per role and keeping track of Current Conversation

“There is a included Colab link in every Doc –There is Change in Prompt(product decision test cases) which is present in the code so every product decision has a different Google Colab link”

Test Cases :

1) High-level Product decisions :

a) Create Transparent Rejection Feedback with Appeals

The debate took 20 rounds to complete :

Testcase1a→ refer to Doc for Debate exchanges, give final recommendation and summary of debate ,There is a included Colab link –There is Change in Prompt(decision) which is present in code

b) Introduce AI Moderation Layer Pre-Publish

Debate took 20 rounds to complete :

Testcase1b → refer to Doc for Debate exchanges give final recommendation and summary of debate ,There is a included Colab link –There is Change in Prompt(decision) which is present in code

2) Low level Product decisions :

a)Add 'Verify Source' Button on Rejected Images

Debate took only 10 rounds to Complete **:**

Testcase2a —> refer to Doc for Debate exchanges gives final recommendation and summary of debate, There is a included Colab link –There is Change in Prompt(decision) which is present in ,code

b)Throttle Uploads After 3 Rejections in 10 Minutes

Debate took only 8 rounds to Complete **:**

Testcase2b—--> refer to Doc for Debate exchanges gives final recommendation and summary of debate There is a included Colab link –There is Change in Prompt(decision) which is present in code

We observe that **lower-level product decisions concluded in fewer debate rounds** compared to high-level strategic ones. This suggests that our **LLM-powered debate system, backed by role-specific RAG pipelines**, enables quicker consensus when the scope is narrower and more tactical

End-to-End Process Flow :

Initialization Phase

- **Knowledge Base Creation**: Domain knowledge for each role is processed and stored in FAISS
  For PM , Designer , Engineer separate knowledge bases are created separately along with product state context (**knowledge base = Role based knowledge + current product state contex**t) but Moderator **(has knowledge base of all agents and also extra moderation knowledge base**)

- **Retriever Setup**: Vector retrievers are configured for each knowledge base
- **Chain Configuration**: RetrievalQA chains connect retrievers to OpenAI models via OpenRouter
- **Debate Parameters**: Topics, speakers, and debate rules are defined

Debate Execution Phase

- **Topic Selection:** The system selects the current topic from a rotating list of 5 topics
- **Speaker Rotation:** For each round, PM → Designer → Engineer take turns
- **Context Window:** Each agent receives the last 3 conversation turns as context
- **Prompt Construction:** Role-specific instructions + topic focus + uniqueness requirements

RAG Process:

- Query → Vector Embedding → Similarity Search → Knowledge Retrieval → Enhanced Prompt → OpenAI via OpenRouter
- **Uniqueness Check:** Responses are checked for sentence-level uniqueness
- **Conversation Update:** Valid responses are added to conversation history

Moderation Phase

- **Termination Assessment:** After each round, the Moderator evaluates the last 9 turns
- **Decision Logic: Determines** if debate has reached maturity or further discussion would add value
- **Continuation or Conclusion:** Either proceeds to next round or terminates debate
- **Final Summary:** Upon termination, generates comprehensive summary with recommendations

Key Technical Components

1. Agent Structure

- **Role-Specific Agents:** PM, Designer, Engineer with domain-specific knowledge
- **Moderator Agent**: Meta-agent with access to all knowledge and decision-making authority

2. Dynamic Debate Control

- **Flexible Length:** Up to 20 potential rounds, typically fewer based on Moderator decision
- **Topic Rotation:** 5 distinct sub-topics that cycle through the rounds
- **Termination Logic:** Intelligent assessment of debate maturity

3. Uniqueness Enforcement

```
def is_response_unique(response: str, used_set: set) -> bool:
    sentences = response.split('. ')
    return all(sentence.strip().lower() not in used_set or
               len(sentence.strip()) <= 10 for sentence in sentences)
```

- Tracks all previously used sentences
- Allows up to 3 retry attempts for non-unique responses
- Ensures debate progression rather than repetition

4. Context Management

- **Agent Context**: Last 3 contributions (focused, recent context)
- **Moderator Context**: Last 9 contributions (broader context for termination decisions)
- **Summary Context**: Full conversation history (comprehensive context for final summary)

Technical Implementation Highlights

1. Vector Database Efficiency

- In-memory FAISS implementation for fast similarity search
- Local embedding generation to avoid API dependencies
- Optimized chunk size balancing context and specificity

2. Prompt Engineering

- Role-specific instructions enforce perspective
- Topic focusing ensures coherent progression
- Brevity constraints (1-2 sentences) maintain conciseness
- Explicit uniqueness requirements prevent repetition

3. Moderator Intelligence

python

```
def moderator_wants_to_stop(convo_history: list) -> bool:
    prompt = (
        "You are the debate Moderator. Based on the following conversation, do you believe the debate should now stop?\n"
        "Only respond with 'YES' if the discussion has reached maturity or further debate would not add value.\n"
        "Respond 'NO' to continue. If 'YES', briefly explain why.\n\n"
        "Conversation:\n" + "\n".join(convo_history[-9:]) + "\n\n"
        "Should the debate stop? Answer 'YES' or 'NO' (followed by reason if YES)."
    )
    # OpenAI model via OpenRouter makes the meta-decision
    result = moderator_chain.run(prompt).strip()
    return result.upper().startswith("YES")
```

Meta-cognitive assessment of debate progress

- Explicit reasoning for termination decisions
- Autonomous control over debate length

4. OpenAI Model Utilization

- **Response Generation**: Primary use of OpenAI models for generating agent responses
- **Meta-Decisions:** Moderator uses OpenAI for termination decisions and summarization
- **Consistency Management**: Temperature settings tuned for role-appropriate responses
- **Token Optimization:** Careful prompt design to maximize information within context limits
-

System Limitations and Future Improvements

Current Limitations

- **Semantic Understanding**: Lexical rather than semantic uniqueness checking
- **Fixed Roles:** Predetermined agent perspectives
- **Limited Intervention:** Moderator primarily controls termination rather than redirection
- **API Dependency**: Reliance on OpenRouter and OpenAI availability and pricing

Potential Improvements

- **Semantic Uniqueness**: Use embeddings to detect conceptual repetition
- **Dynamic Topics:** Generate sub-topics based on debate progress
- **Real-Time Fact-Checking**: Verify factual claims during debate
- **Model Flexibility:** Support for switching between different LLM providers
- **Local LLM Option:** Fallback to local models when API is unavailable

Conclusion

The LLM-powered Debate Moderator demonstrates sophisticated RAG implementation with dynamic debate control. The system leverages vector databases, OpenAI models via OpenRouter, contextual retrieval, and intelligent moderation to create structured, non-repetitive multi-agent debates that generate meaningful insights on product decisions.