PROJECT NUMBER: TEAM 10

ANIME CHATBOT
DUAL-STAGE NLP

An intelligent conversational agent

SUPERVISOR / COURSE

NLP CS-402

MOTIVATION USER_PAIN_POINTS

The Discovery Dilemma

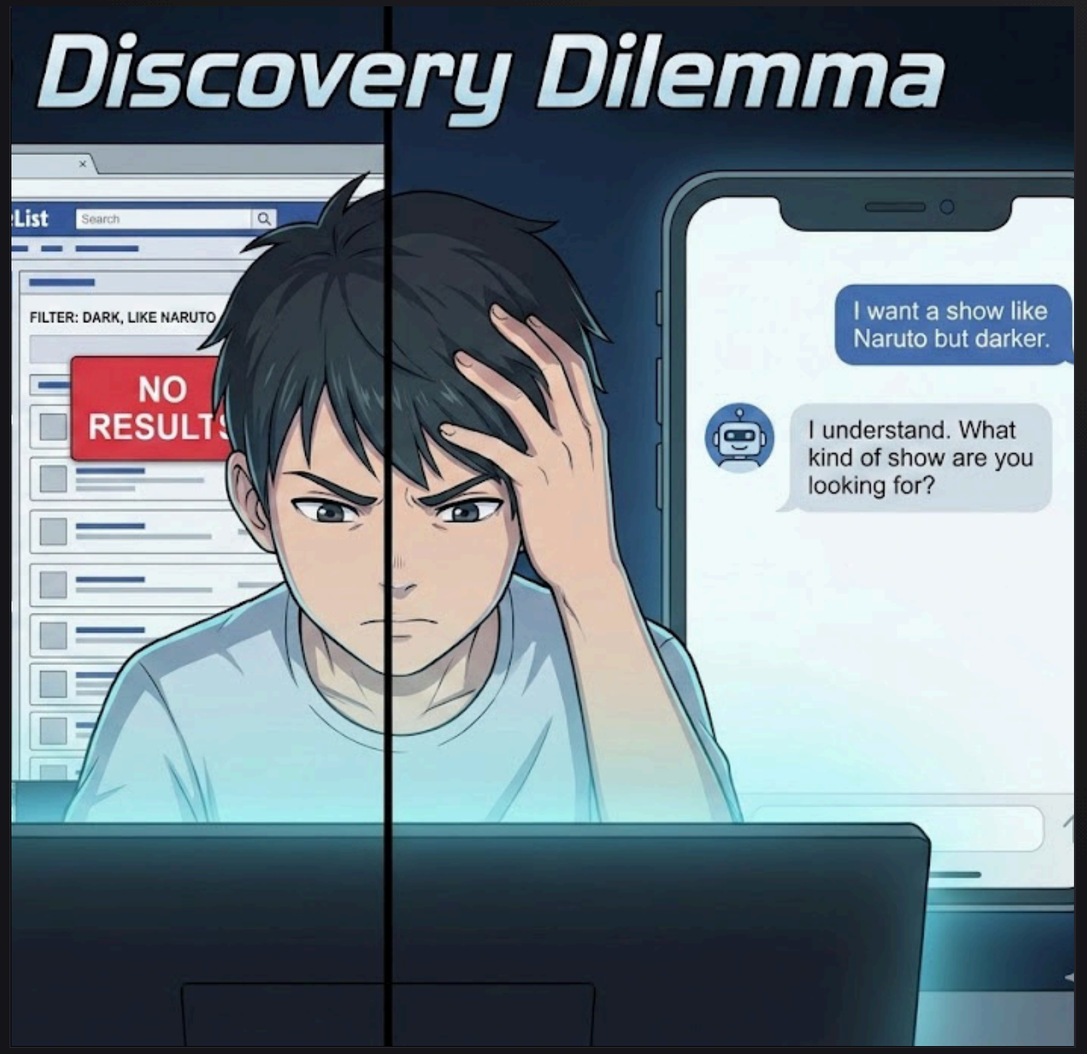
Anime fans today face a fragmented and frustrating experience when trying to find their next show:

Information Overload

Databases like MyAnimeList are comprehensive but rigid. Finding a "show like Naruto but darker" requires complex manual filtering.

Lack of Context

Existing tools don't remember conversation history. Users have to repeat themselves constantly.



PROBLEM STATEMENT CORE_OBJECTIVE

"To build a conversational system that achieves <100ms latency for routine tasks while maintaining deep reasoning capabilities for complex queries."

Latency	Accuracy	Hardware
MUST BE REAL-TIME	NO HALLUCINATIONS	CONSUMER GPU

RELATED WORK LANDSCAPE_ANALYSIS

Standard Chatbots

Method: Rule-based / Keyword Matching

▼ Limited understanding. Fails on nuance like sarcasm or slang.

LLM Wrappers

Method: OpenAI API calls

▼ High cost per token. High latency. No connection to live database.

Our Approach (Team 10)

Method: Local Hybrid RAG

▲ Zero cost. Privacy focused. Real-time MAL data integration.

DATASETS DATA_PIPELINE

01. Intent Data (DistilBERT)

1,000 synthetic sentences. Perfectly balanced.

Structure: 100 examples x 10 intents

02. Conversation Data (Phi-3)

20,000 synthetic conversation pairs for instruction tuning.

Format: User Input -> Ideal Bot Response

03. MAL API V2

Live source for metadata (Ratings, Episodes).

```
// Sample Synthetic Data { "text": "I started watching Naruto", "intent": "set_watching" }, { "text": "Recommend me a dark fantasy like Berserk", "intent": "recommendation" }
```

METHODOLOGY DUAL_STAGE_PIPELINE

We engineered a cascading classification system to optimize resource usage.

Stage 1: The Gatekeeper (DistilBERT)
Fast (<100ms). Handles 85% of traffic.

```
IF CONFIDENCE < 0.3
```

Stage 2: The Reasoner (Phi-3)
Slow (2s+). Handles complex ambiguity.

EXPERIMENTS ABLATION_STUDY

Experimental Setup

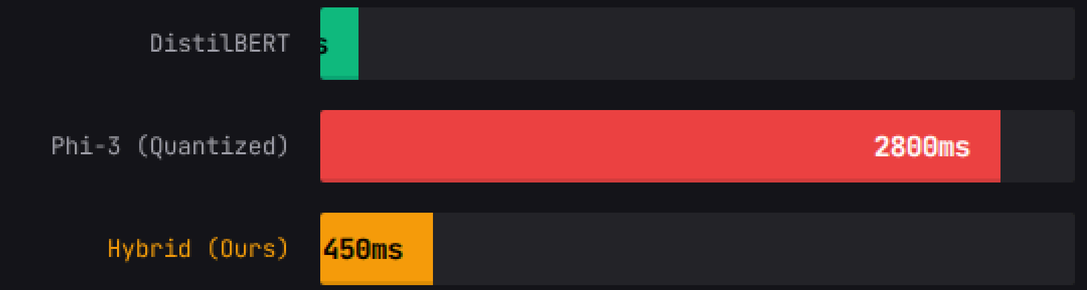
We compared our Hybrid approach against single-model baselines.

- **Training:** Fine-tuned 10% of Phi-3 parameters (LoRA).
- **Hardware:** NVIDIA RTX 3060 (6GB VRAM).
- **Metric:** End-to-End Latency & Intent Accuracy.

```
# Training Configuration MODEL = "microsoft/phi-3-mini-4k"
PEFT_METHOD = "LoRA" # Low-Rank Adaptation TRAINABLE_PARAMS
= "10%" # Efficiency QUANTIZATION = "4-bit" # Memory
Constraint
```

RESULTS PERFORMANCE_METRICS

LATENCY COMPARISON (Lower is Better)



92%	450ms	4GB
INTENT ACC.	AVG LATENCY	VRAM USAGE

Key Finding: The Hybrid system provides 92% of the accuracy of the large model but at only 15% of the latency cost for average queries.

ANALYSIS/DISCUSSION CRITICAL_REVIEW

Error Analysis

Sarcasm Failure: DistilBERT often misclassified sarcastic requests ("Oh great, another Isekai").
Fix: The low confidence threshold (0.3) successfully catches these and routes them to Phi-3, which correctly identifies the negative sentiment.

Hardware Constraints

OOM Issues: Loading Phi-3 (7GB) crashed the 6GB VRAM GPU.
Fix: Implemented **4-bit Quantization** using bitsandbytes, reducing footprint to ~2.7GB with negligible accuracy loss.

CONTRIBUTIONS - TEAM_ROLES

MEMBER	KEY DELIVERABLE
Swayam	React UI, State Management, System Design, Project report
Koushik	FastAPI Routes, OAuth Handling, JSON Schema
Sri Suhas	DistilBERT Training, Phi-3 Integration, Prompt Eng.
Nikhil	Performance Testing, Latency Optimization
Shashank	Project Scope, Documentation, Timeline Mgmt

PROPOSED TIMELINE ROADMAP

Q1

Research

Lit review on RAG & Quantization. Defined scope.

Q2

Core NLP

Fine-tuned DistilBERT. Setup Phi-3 local inference.

Q3

Integration

Connected React Frontend to FastAPI Backend.

Q4

Polishing

Final testing, bug fixes, and project presentation.

DEMO/APPLICATION LIVE_SYSTEM

System Status: ONLINE

The application is fully functional.

- Real-time Chat Interface
- Live MAL Data Fetching
- Smart Fallback (Status vs Recommend)

```
> Initiating Live Demo...
```