

Fynd AI Intern – Take Home Assignment Report

Name: Arjun Maurya

Role: AI Intern (Take Home Assignment)

Tech Stack: Python, FastAPI, SQLite, OpenRouter LLM, HTML, JavaScript, Vercel, Render

Task 1 – Rating Prediction via Prompting

Objective

The goal of Task 1 was to predict star ratings (1–5) from user review text using Large Language Models (LLMs) through prompt engineering, without training any model.

Dataset

- Dataset: Yelp Reviews
 - Fields used:
 - **text** – user review
 - **stars** – actual rating (ground truth)
-

Approach

Instead of training a model, I used **prompt-based prediction**.

I designed **three different prompt versions** and compared their performance.

Each prompt asked the LLM to:

- Read the review text
 - Predict the rating from **1 to 5**
 - Return output in **structured JSON format**
-

Prompt Versions

Prompt v1 – Basic Prompt

- Simple instruction to predict rating
- No formatting rules

Result:

- Inconsistent outputs
 - Sometimes returned text instead of a number
-

Prompt v2 – Structured Prompt

- Clearly instructed rating range (1–5)
- Asked for JSON output

Result:

- Improved consistency
 - Still failed occasionally with invalid JSON
-

Prompt v3 – Strict JSON Prompt

- Explicitly asked for **ONLY valid JSON**
- No explanation or markdown allowed

Result:

- Best accuracy
 - Almost zero formatting errors
 - Reliable structured output
-

Evaluation Metrics

- Accuracy (comparison with actual star rating)
 - JSON validity
 - Consistency of responses
-

Conclusion (Task 1)

Prompt v3 performed the best due to:

- Strict formatting rules
- Clear instructions
- Reduced ambiguity for the LLM

This shows how **prompt engineering alone** can significantly improve LLM reliability.

Task 2 – AI Feedback System

Objective

Build a complete AI-powered feedback system where:

- Users submit reviews
 - AI generates a polite response
 - Admin receives AI-generated insights
-

System Architecture

Frontend (HTML + JS)

|

v

Backend (FastAPI)

|

v

SQLite Database

|

v

OpenRouter LLM

Features Implemented

1. User Dashboard

- User submits:
 - Rating (1–5)
 - Review text
 - Backend:
 - Stores review in database
 - Sends review to LLM
 - Returns AI-generated reply
-

2. AI Processing

Two LLM prompts are used:

User Reply Prompt

- Generates a polite, friendly response for the user

Admin Analysis Prompt

- Returns structured JSON with:
 - Summary of review
 - Recommended admin action
-

3. Admin Dashboard

- Fetches all reviews from backend
 - Displays:
 - Rating
 - Review text
 - AI-generated summary
 - Suggested action
-

Backend Implementation

- Framework: **FastAPI**
- Database: **SQLite**
- API Endpoints:
 - `POST /submit-review`
 - `GET /admin/reviews`

- Environment variables handled securely via `.env`
-

Deployment

- **Backend:** Render
- **Frontend (User + Admin):** Vercel

All services are publicly accessible and fully functional.

Error Handling

- Empty review validation
 - JSON extraction from LLM responses
 - API failure handling
 - Secure API key management
-

Limitations

- Depends on external LLM API availability
 - SQLite is suitable for demo, not large-scale production
 - Basic frontend UI (focus was backend & AI logic)
-

Conclusion

This project demonstrates:

- Effective use of prompt engineering
- End-to-end AI system design
- Clean API architecture
- Real-world deployment practices

The assignment fulfills all requirements mentioned in the problem statement.

Links

- **GitHub**
Repository: <https://github.com/Arjunmauryaaa/fynd-ai-assignment/tree/main>
- **Backend URL:** <https://fynd-ai-assignment.onrender.com>
- **User Dashboard:** <https://fynd-ai-assignment-rkuw.vercel.app/>
- **Admin Dashboard:** <https://fynd-ai-assignment-iyas.vercel.app/admin.html>