Fynd AI Intern – Take Home Assessment 2.0

Deliverables

You must submit:

1. A GitHub Repository (mandatory)

The repository must contain:

- The Python notebook for **Task 1**
- Application code for **Task 2**
- Any supporting files (schemas, prompts, configs, etc.)
- **Deployment links for both dashboards (mandatory)**

2. A Short Report

A short report (PDF preferred) summarising:

- Your overall approach
- Design and architecture decisions
- Prompt iterations and improvements
- Evaluation methodology and results (Task 1)
- System behaviour, trade-offs, and limitations (Task 2)

3. Deployed Dashboards (Mandatory)

Both **User** and **Admin** dashboards must be:

- Fully deployed
- Publicly accessible via URLs
- Functional without local setup

Note: Faster completion is viewed positively.

LLM Usage

You may use **any LLM** you prefer.

Free options include:

- Gemini API (free tier)
- OpenRouter (free open-source models)

Choose whichever works best for your experiments.

TASK 1 – Rating Prediction via Prompting

Design prompts that classify Yelp reviews into **1–5 star ratings**, returning **structured JSON**.

Dataset

Use the Yelp Reviews dataset from Kaggle:
https://www.kaggle.com/datasets/omkarsabnis/yelp-reviews-dataset
You may sample a subset for efficiency.

Output Format

```
{
 "predicted_stars": 4,
 "explanation": "Brief reasoning for the assigned rating."
}
```

Requirements

- Implement **at least 3 different prompting approaches** (your own design).
- Evaluate how each approach affects:
    - Accuracy (Actual vs Predicted)
    - JSON validity rate
    - Reliability and consistency

You must:

- Clearly show **each prompt version**
- Explain **why and how** you improved or changed each prompt
- Evaluate on a sampled dataset (**~200 rows recommended**)
- Provide:
    - A comparison table

- A short discussion of results and trade-offs

TASK 2 – Two-Dashboard AI Feedback System (Web-Based)

Build a **production-style web application** with two dashboards.
Both dashboards must be fully deployed.

Important Constraints (Read Carefully)

- **Streamlit, HuggingFace Spaces, Gradio, or notebook-based apps are NOT allowed**
- The system **must be a real web application.**
- Deployment **must be on platforms like Vercel, Render, etc.**

Submissions violating these constraints will be **rejected**.

A. User Dashboard (Public-Facing)

Users should be able to:

- Select a star rating (1–5)
- Write a short review
- Submit the review

On submission:

- An **AI-generated response** must be shown to the user
- The submission must be **stored via a backend service**
- The user should see a clear success / error state

B. Admin Dashboard (Internal-Facing)

The Admin Dashboard must display a **live-updating or auto-refreshing list** of all submissions, including:

- User rating

- User review
- AI-generated summary
- AI-suggested recommended actions

Additionally:

- The Admin Dashboard may include any analytics you deem useful (e.g., filters, counts by rating, trends, etc.)

System Requirements

- Both dashboards must:
  - Be **web-based**
  - Be deployed on **Vercel / Render or similar platforms**
- Both dashboards must:
  - Read from and write to the **same persistent data source**
- LLMs must be used for:
  - Review summarisation
  - Recommended next actions
  - User-facing responses

Technical Requirements (Mandatory)

- All LLM calls must be **server-side**
  (No client-side API calls to LLMs)
- Backend must expose **clear API endpoints**
- Request and response payloads must use **explicit JSON schemas**
- The system must handle:
  - Empty reviews
  - Long reviews
  - LLM or API failures gracefully

Deployment Requirements

You must provide:

- Public **User Dashboard URL**
- Public **Admin Dashboard URL**

Deployments must:

- Load successfully
- Persist data across refreshes
- Function without manual intervention

Final Submission Format

Submit the following:
GitHub Repository (must include the notebook):
Report PDF Link:
User Dashboard URL:
Admin Dashboard URL: