AI Agent Assignment

SECTION 1:

Title: Daily Headline Explainer Agent

In the modern digital world, people are constantly exposed to a large volume of news from multiple sources, making it difficult to quickly understand the importance and context of each headline. News headlines are often brief and assume prior knowledge, which can confuse readers or lead to misinterpretation. The Daily Headline Explainer Agent addresses this challenge by automatically analyzing news headlines from a large newspaper dataset and converting them into clear, structured explanations. The agent identifies the topic, key entities, and summarizes the event in simple language. Additionally, it provides historical background to help users understand the broader context of the news. This application is particularly useful for students, professionals, and casual readers who want to stay informed without reading full articles.

SECTION 2: PROBLEM FRAMING

2.1 What problem does your AI Agent solve?

Modern news platforms publish thousands of articles daily, making it difficult for users to quickly grasp the meaning and importance of each headline. Headlines are often short, context-heavy, and assume prior knowledge, which can lead to confusion or misinformation. This agent solves the problem of quickly understanding news by converting raw headlines into clear, contextual explanations.

2.2 Why is this agent useful?

The agent delivers value by reducing information overload and saving time for users. Instead of reading full articles, users receive a structured explanation that includes the topic, key entities, a simple summary, and historical background, enabling faster and more accurate understanding of current events.

2.3 Who is the target user?

The target users include college students staying updated with current affairs, professionals who want quick news insights during busy schedules, competitive exam aspirants, and casual readers who want simplified explanations without deep political or economic knowledge.

SECTION 3: 4-LAYER PROMPT DESIGN

Before designing the agent, a basic research phase was conducted to understand what tools and resources would best suit the problem. A BBC News dataset was identified, which provides structured news headlines along with links. Next, both open-source LLMs and API-based models were explored to evaluate feasibility on limited hardware. Due to performance and deployment constraints, an API-based LLM (Groq) was selected.

The design of the Daily Headline Explainer Agent began with an initial exploration of how prompt-based agents can be structured to reason step by step instead of producing unstructured answers. To achieve this, a **4-layer prompt architecture** was adopted, where each layer has a clear responsibility and builds upon the previous one.

The first layer designed was **Input Understanding**.
This layer focuses on interpreting the user's message and removing ambiguity. Since news headlines are often short and context-heavy, the agent needs to clearly identify what the headline refers to before doing any analysis.

Prompt used:

"You are an AI agent that receives a user message.
Identify:

1. The news headline mentioned
2. The topic (economy, politics, sports, etc.)
3. The key entity involved
4. A short summary of the incident."

This prompt ensures the agent extracts the core meaning of the news by identifying the subject, category, and main actors involved. It acts as the foundation for all further reasoning.

Next, the State Tracker layer was designed.
The purpose of this layer is to help the agent remember important context across

steps, similar to short-term memory. While the project does not use long-term memory storage, contextual information is preserved implicitly across prompts.

Prompt used:

“Store important context from the conversation such as:
– Country
– Institution
– Topic
Reuse this context for follow-up questions unless the user changes it.”

This allows consistency in the explanation, especially when generating summaries and historical background, without repeatedly reinterpreting the same information.

The third layer introduced was the Task Planner.
This layer decides *how the agent should think* before generating the final response. Instead of directly producing an answer, the agent is guided through a structured reasoning process.

Prompt used:

“Before answering, follow these steps:

1. Think about background concepts needed
2. Describe what happened in the headline
3. Think about the impact on common people
   Do not add opinions or predictions.”

This step enforces neutral, factual reasoning and prevents speculative or biased outputs. At this stage, prompt chaining is applied, where the outputs of Input Understanding and State Tracking feed into the Task Planner. The agent follows a linear chain rather than branching, keeping complexity manageable.

Finally, the Output Generator layer was designed to control how the answer is presented to the user.
This layer focuses on clarity, formatting, and readability.

Prompt used:

“Generate the final answer in the following format:
– Headline
– Topic (economy, politics, sports, etc.)
– Key entity involved
– A short summary in simple language.”

Initially, this output only contained a short explanation. Later, the design was evolved to include a Background / History section, providing users with brief historical context related to the news.

All four prompts were used in a single ChatGPT session to create a basic Python script was generated to automate the process using an LLM API.

After the backend logic was finalized, attention was shifted to user experience. Using Lovable AI, a frontend prompt was created to design a purple-themed interface. The UI was designed with the following features:

- All headlines from the dataset are displayed in a scrollable sidebar
- The analyzed output appears on the right panel

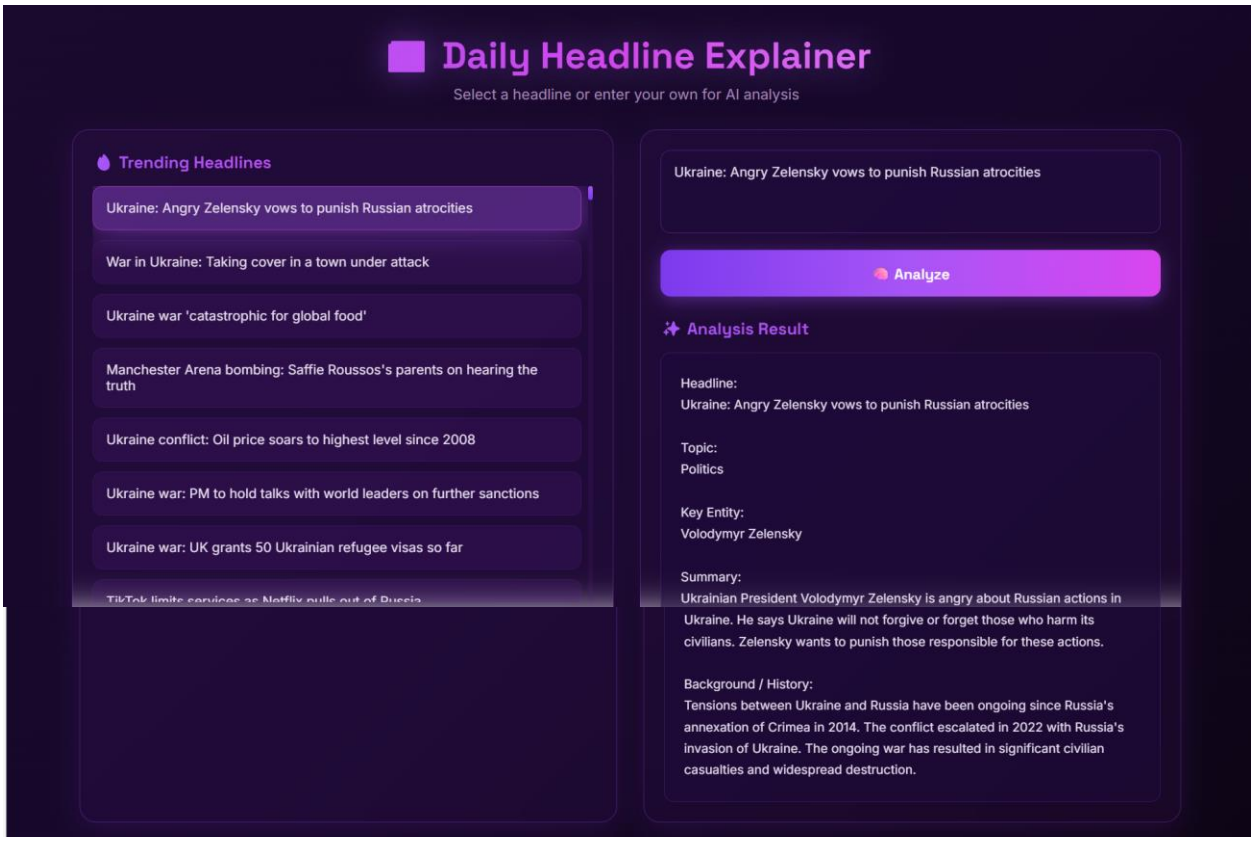
Finally, the frontend developed using HTML, CSS, and JavaScript was connected to the Python backend by prompting ChatGPT to use Flask as a lightweight web framework. This integration enabled seamless communication between the user interface and the AI logic. The final agent follows a chained prompt architecture, ensuring structured reasoning, consistent output, and a clear separation of responsibilities across the different prompt layers.

SECTION 4: CHATGPT EXPLORATION LOG

Attempt	Prompt Variant	What Happened	What You Changed	Why You Changed It
1	“Generate a simple explanation for this news headline”	The output was very generic and lacked structure.	Introduced a structured prompt with fields like topic, entity, and summary.	To make the output clearer and more consistent.
2	“Identify topic and summarize the headline”	Topic detection worked, but context was missing.	Added link from the dataset as additional input.	To improve understanding and reduce ambiguity.
3	“Explain the headline step by step”	Output contained opinions and assumptions.	Added explicit instructions to avoid opinions and predictions.	To maintain neutrality and factual correctness.
5	“Add background/history of this news”	Background information was mixed with the main summary.	Created a separate Background / History section in the prompt.	To clearly separate current events from historical context.
7	“Create a frontend for this agent”	Frontend design was basic and unstructured.	Used Lovable AI to generate a themed UI with a scrollable sidebar.	To improve usability and visual clarity.

8	“Connect frontend and backend”	Manual integration approach was unclear.	Prompted ChatGPT to use Flask for backend integration.	To enable smooth communication between the UI and Python logic.

SECTION 5: OUTPUT TESTS



This output shows the Daily Headline Explainer interface, where users select a news headline from a scrollable list or enter their own. After clicking Analyze, the system displays a structured explanation on the right, including the headline, topic, key entity, a simple summary, and relevant background history for quick understanding.

SECTION 6: REFLECTION

6.1 What was the hardest part of this assignment?

The hardest part of this assignment was choosing an appropriate large language model. Initially, I experimented with open-source models like LLaMA and Phi from Hugging Face, but they required large downloads and long execution times due to hardware constraints. Managing memory limitations and slow local inference made deployment difficult. As a result, I shifted to using the Groq API, which provided faster responses without the need for heavy local computation. This experience highlighted the importance of aligning model selection with available system resources and project constraints.

6.2 What part did you enjoy the most?

I enjoyed every part of the learning process, especially experimenting with prompt design and seeing how small changes in prompts significantly affected the output quality. Designing the 4-layer prompt structure was particularly interesting. Building the complete system from idea to working interface was also very satisfying.

6.3 If given more time, what would you improve or add?

If given more time, I would add real-time news fetching instead of relying only on a static dataset. I would also implement user personalization features, such as topic-based filtering. Improving response speed and adding multilingual support would further enhance usability.

6.4 What did you learn about ChatGPT or prompt design?

I learned that well-structured prompts greatly improve clarity, consistency, and reliability of outputs. Breaking a task into multiple prompt layers helps guide the model's reasoning more effectively. Prompt chaining is especially useful for complex tasks that require structured thinking.

6.5 Did you ever feel stuck? How did you handle it?

Yes, I felt stuck multiple times, especially when dealing with model loading errors, API issues, and hardware limitations. I handled this by breaking the problem into smaller parts, testing each component separately, and iterating on prompts. Using experimentation and trying alternative approaches helped me move forward.

SECTION 7: HACK VALUE

This project implements a structured, multi-layer prompt architecture rather than relying on a single, monolithic prompt. Additionally, a custom frontend was designed to allow users to interactively select headlines or enter their own, enhancing usability beyond a simple command-line solution.

DONE BY: Nandana Thachilath.M
Email: nanduthachilath2004@gmail.com