Ak Intern

I have carefully gone through the entire assignment document. Here’s a structured breakdown:

**Assignment Overview**

You need to build a **News Summarization and Text-to-Speech (TTS) application** that:

* Extracts key details from **at least 10 news articles** about a company.
* Performs **sentiment analysis** (Positive, Negative, or Neutral).
* Conducts a **comparative sentiment analysis** of the articles.
* Converts the summarized content into **Hindi speech** using a TTS model.
* Provides a **web-based UI** using **Streamlit or Gradio**.
* Uses **APIs** for communication between frontend and backend.
* Deploys the application on **Hugging Face Spaces**.

**Main Requirements**

1. **News Extraction**
   * Scrape at least **10 unique** news articles related to the company.
   * Use **BeautifulSoup** (no JavaScript-heavy pages).
   * Extract **title, summary, sentiment, topics**.
2. **Sentiment Analysis**
   * Categorize articles as **Positive, Negative, or Neutral**.
   * Possibly use an **NLP model** (like VADER, TextBlob, or transformers-based models).
3. **Comparative Analysis**
   * Show **sentiment distribution** across articles.
   * Highlight **differences in news coverage**.
   * Identify **common and unique topics** between articles.
4. **Text-to-Speech (TTS)**
   * Convert the sentiment summary into **Hindi speech**.
   * Use an **open-source TTS model** (like VITS, Festival, or Google TTS).
5. **User Interface**
   * Implement a **simple UI** using **Streamlit or Gradio**.
   * Users should be able to **input a company name** and fetch results.
6. **API Development**
   * Ensure **frontend communicates with backend via APIs**.
   * Possibly use **FastAPI or Flask**.
7. **Deployment**
   * Deploy the application on **Hugging Face Spaces**.
8. **Documentation**
   * Submit a **detailed README** with:
     + **Setup instructions**
     + **Model details**
     + **API usage**
     + **Assumptions & limitations**

**Expected Input & Output**

**Input**

* A **company name** entered by the user.

**Output (Structured Report)**

* **Articles:**
  + **Title**
  + **Summary**
  + **Sentiment** (Positive/Negative/Neutral)
  + **Topics**
* **Comparative Analysis**
  + Sentiment distribution across articles.
  + Differences in coverage.
  + Common vs. unique topics.
* **Final Sentiment Summary**
  + Overall sentiment trend.
* **Hindi TTS Output**
  + **Playable Hindi audio** summarizing the sentiment.

**Example Output**

A JSON structure like:

json

CopyEdit

{

"Company": "Tesla",

"Articles": [

{

"Title": "Tesla's New Model Breaks Sales Records",

"Summary": "Tesla's latest EV sees record sales in Q3...",

"Sentiment": "Positive",

"Topics": ["Electric Vehicles", "Stock Market", "Innovation"]

},

{

"Title": "Regulatory Scrutiny on Tesla's Self-Driving Tech",

"Summary": "Regulators have raised concerns over Tesla’s self-driving software...",

"Sentiment": "Negative",

"Topics": ["Regulations", "Autonomous Vehicles"]

}

],

"Comparative Sentiment Score": {

"Sentiment Distribution": {

"Positive": 1,

"Negative": 1,

"Neutral": 0

},

"Coverage Differences": [

{

"Comparison": "Article 1 highlights Tesla's strong sales, while Article 2 discusses regulatory issues.",

"Impact": "The first article boosts confidence in Tesla's market growth, while the second raises concerns about future regulatory hurdles."

}

],

"Topic Overlap": {

"Common Topics": ["Electric Vehicles"],

"Unique Topics in Article 1": ["Stock Market", "Innovation"],

"Unique Topics in Article 2": ["Regulations", "Autonomous Vehicles"]

}

},

"Final Sentiment Analysis": "Tesla’s latest news coverage is mostly positive. Potential stock growth expected.",

"Audio": "[Play Hindi Speech]"

}

**Project Structure**

Your **GitHub repository** must include:

* app.py → Main script
* requirements.txt → Dependencies
* README.md → Setup & instructions
* utils.py → Helper functions (e.g., web scraping, text processing)
* api.py → API endpoints
* Hugging Face Spaces **deployment link**.

**Evaluation Criteria**

✅ **Correctness** → Extracts & processes info accurately  
✅ **Efficiency** → Optimized performance  
✅ **Robustness** → Handles errors & edge cases  
✅ **Deployment** → Working app on Hugging Face Spaces  
✅ **Code Quality** → Follows **PEP8**, well-structured, documented