News Summarization and Text-to-Speech Application

1. Project Setup

Steps to install and run the application.

Prerequisites

- Python 3.10 or higher.
- Pip (Python package manager).

Installation

```
project/
│
├── api.py              # FastAPI backend
├── utils.py            # Utility functions
├── app.py              # Streamlit frontend
└── requirements.txt    # Dependencies
└── config.py           # setup file to create instances of the required API clients.
```

1. Clone the repository:

   git clone https://github.com/SriArunM/News_Summarization.git

   cd News_Summarization

2. Install dependencies:

   pip install -r requirements.txt

3. Set up API keys:

   Obtain API keys for:

   - Hugging Face (for Inference API) ---> Due to less computational resources
   - OpenRouter (for qwq-32B)
   - Google Gemini (Gemma-27B)
   - Groq API key (llama3.3-70B-versatile)

   - Add the keys to the .env file

Running the Application

1. Start the FastAPI backend:

   python api.py

   The backend will be available at http://127.0.0.1:8000.

News Summarization and Text-to-Speech Application

2. Start the Streamlit frontend:

   streamlit run app.py

   The frontend will be available at http://localhost:8501.

---

2. Model Details

Explanation of models used for summarization, sentiment analysis, and TTS.

Summarization

- **Model**: facebook/bart-large-cnn (Hugging Face).
- **Purpose**: Summarizes long articles into concise summaries.
- **Fallback**: Google GenAI (gemma-3-27b-it) if Hugging Face fails.

Sentiment Analysis

- **Primary Model**: Google GenAI (gemma-3-27b-it).
- **Purpose**: Analyzes the sentiment of articles (Positive, Negative, Neutral).
- **Fallback**: Hugging Face (finiteautomata/bertweet-base-sentiment-analysis) if Google GenAI fails.

Text-to-Speech (TTS)

- **Primary Model**: gTTS (Google Text-to-Speech).
- **Purpose**: Converts Hindi text to speech.
- **Fallback**: Hugging Face MMS-TTS (facebook/mms-tts-hin) if gTTS fails.

Topic Extraction

- **Primary Model**: Groq (llama-3.3-70b-versatile).
- **Purpose**: Extracts key topics from articles.
- **Fallback**: Google GenAI (gemma-3-27b-it) if Groq fails.

---

3. API Development

Details on how the APIs are being used and how to access them.

FastAPI Backend

- **Base URL**: http://127.0.0.1:8000
- **Endpoints**:
    1. **POST /analyze-news**:
        - **Purpose**: Fetches and analyzes news articles for a given company.

News Summarization and Text-to-Speech Application

- **Request Body**:


```
{
  "company_name": "Tesla",
}
```


- **Response**:


```
{
  "Company": "Tesla",
  "Articles": [
    {
      "URL":  "https://example.com/article1",
      "Title": "Tesla News",
      "Summary": "Summary of the article.",
      "Summary(in Hindi)": "लेख का सारांश।",
      "Sentiment": "Positive",
      "Reason(for Sentiment)": "The article highlights positive developments.",
      "Topics": ["Electric Vehicles", "Innovation"]
    }
  ],
  "Comparative Sentiment Score": {
    "Sentiment Distribution": {
    "Positive": 5,
      "Negative": 3,
      "Neutral": 2
    },
    "Total Sentiments": 10
  },
  "Final Sentiment Analysis": "Overall sentiment is positive.",
  "Audio": "output.mp3"
}
```


Accessing APIs via Postman

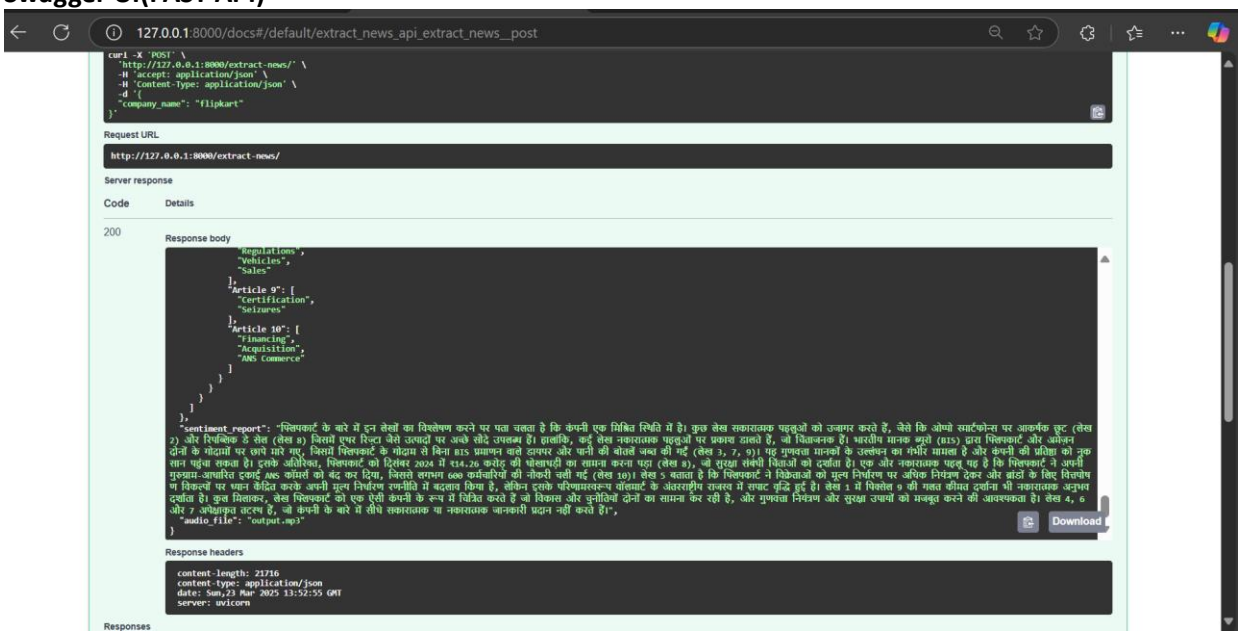
News Summarization and Text-to-Speech Application

1. Open Postman.
2. Set the request type to POST.
3. Enter the URL: http://127.0.0.1:8000/analyze-news.
4. Set the body to raw and JSON.
5. Add the request body:

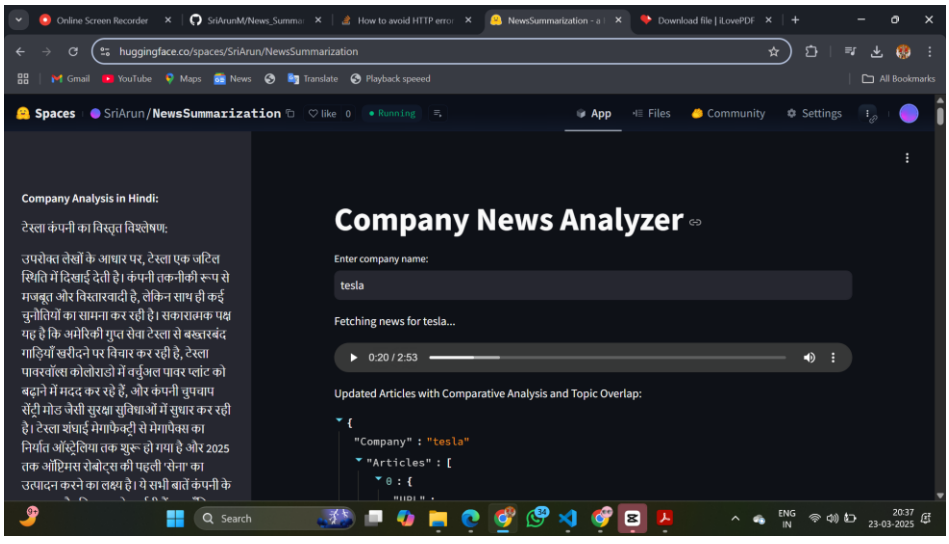
```
{
  "company_name": "Tesla",
}
```

6. Click **Send**.

Swagger UI(FAST API)



Streamlit (from hugging face spaces)



News Summarization and Text-to-Speech Application

4. API Usage

Details on third-party APIs used in the project.

Third-Party APIs

1. **Hugging Face**:
    - **Purpose**: Summarization and sentiment analysis.
    - **Integration**: Used via the InferenceClient .
2. **Google Gemini**:
    - **Purpose**: Summarization, sentiment analysis, and topic extraction.
    - **Integration**: Used via the genai.Client .
3. **OpenRouter**:
    - **Purpose**: Fallback for comparative analysis.
    - **Integration**: Used via the OpenAI client.
4. **MyMemory Translation API**:
    - **Purpose**: Translates summaries into Hindi.
    - **Integration**: Used via the translate.Translator .
5. **gTTS**:
    - **Purpose**: Converts Hindi text to speech.
    - **Integration**: Used via the gTTS library .
6. **Groq**:
    - **Purpose**: Extract Topics from articles.

News Summarization and Text-to-Speech Application

- **Integration**: Used via the langchain ChatGroq.

---

5. Assumptions & Limitations

Assumptions

1. **API Keys**:
   - Valid API keys are available for all third-party services.
   - Rate limits for APIs are not exceeded during runtime.
2. **Input Data**:
   - The company name provided by the user is valid and returns relevant news articles.
3. **Fallback Mechanisms**:
   - If one API fails, the fallback API will provide acceptable results.

Limitations

1. **Rate Limits**:
   - Groq and Hugging Face APIs have rate limits that may affect performance.
2. **Error Handling**:
   - Some edge cases (e.g., empty articles or invalid URLs) may not be fully handled.
3. **Dependency on Third-Party APIs**:
   - The application relies heavily on third-party APIs, which may experience downtime or changes.
4. **Inability to extract JavaScript-Rendered Content**:
   - The current implementation only extracts static HTML content and does not handle JavaScript-rendered content, limiting its ability to process dynamically loaded data on modern websites.

6. Future Improvements

1. **Caching**:
   - Implement caching for API responses to reduce rate limit issues.
2. **Enhanced Error Handling**:
   - Add more robust error handling for edge cases.
3. **Multi-Language Support**:
   - Extend TTS and translation features to support more languages.
4. **Local Models**:

   - Use local models for summarization and sentiment analysis to reduce dependency on third-party APIs.

5. **User Interface**:
   - Improve the Streamlit frontend with better visualization and interactivity.

---

7. Conclusion

This project provides a comprehensive solution for analyzing news articles related to a company. It uses state-of-the-art models for summarization, sentiment analysis, and TTS, with fallback mechanisms to ensure reliability. The FastAPI backend and Streamlit frontend make it easy to use and integrate into larger systems.