

LinguaAI

I. Overview of the Project

This project is a sophisticated Natural Language Processing (NLP) API that provides text summarisation and sentiment analysis functionalities. The backend is built using FastAPI, a modern web framework for building APIs with Python, while the front end is developed using React, a popular JavaScript library for building user interfaces. The core functionality relies on pre-trained models from Hugging Face's Transformers library, which offers state-of-the-art NLP capabilities. This project is designed for developers who need efficient and accurate NLP functionalities in their applications.

Key Features

- **Text Summarization:** Condenses long pieces of text into concise summaries while preserving the main ideas.
- **Sentiment Analysis:** Determines the text's sentiment (positive, negative, neutral).
- **Secure and Reliable:** Implements API key authentication, input sanitisation, and other security measures to ensure safe usage.
- **Scalable and Performant:** Uses lightweight, efficient models and asynchronous programming to handle multiple requests quickly and efficiently.

Technologies Used

- **FastAPI:** For building the web API, chosen for its speed and ease of use.
- **React:** Provides a dynamic and responsive user interface with a modern front-end development experience.
- **Hugging Face Transformers:** For NLP model implementations, providing robust pre-trained models for various NLP tasks.
- **Docker:** For containerising the application, ensuring consistency across different environments and simplifying deployment.
- **Pydantic:** For data validation and settings management, ensuring robust and error-free configurations and data handling.
- **Bleach:** For input sanitisation, protecting against injection attacks and ensuring input data safety.

II. Setting Up and Running the Service

A]. On Local Machine

Prerequisites

Before setting up the project, ensure you have the following installed:

- Python 3.11 or later
- Node.js and npm (for React)
- Docker (optional but recommended for containerized deployment)

Backend Setup

1. Navigate to **Language_Model_For_Local/backend/lmapp** directory:

```
bash
```

```
cd Language_Model_For_Local/backend/lmapp
```

2. Create and Activate a Virtual Environment:

```
bash
```

```
python -m venv venv
source venv/bin/activate # On Windows use `venv\Scripts\activate`
```

3. Install Backend Dependencies:

```
bash
```

```
pip install -r requirements.txt
```

4. Install specific versions of PyTorch, torchvision, and torchaudio using pip with the given index URL.

Command:

pip install torch==2.2.2 torchvision==0.17.2 torchaudio==2.2.2 --index-url <https://download.pytorch.org/whl/cpu>

```
Language_Model_For_Local\backend>pip install torch==2.2.2 torchvision==0.17.2 torchaudio==2.2.2 --index-url https://download.pytorch.org/whl/cpu
```

5. Configure Environment Variables: Create a .env file in the root directory and add the following variables:

```
env
```

```
API_KEY=your_api_key_here
```

6. Run the Backend Server:

```
bash
```

```
uvicorn main:app --reload --host 0.0.0.0 --port 8000
```

Frontend Setup

1. Navigate to Language_Model_For_Local/frontend/lmapp directory:

```
bash
```

```
cd Language_Model_For_Local/frontend/lmapp
```

2. Install Frontend Dependencies:

```
bash
```

```
npm install
```

3. Start the React Development Server:

```
bash
```

```
npm start
```

B]. Build Docker Images

1. Clone the Repository: Navigate to **Language_Model_For_Docker/** directory:

```
bash
```

```
cd Language_Model_For_Docker
```

2. **Build Docker Images and Start the Containers:**

```
bash
```

```
docker-compose up --build
```

III. API Setup, Configuration, and Usage Guide

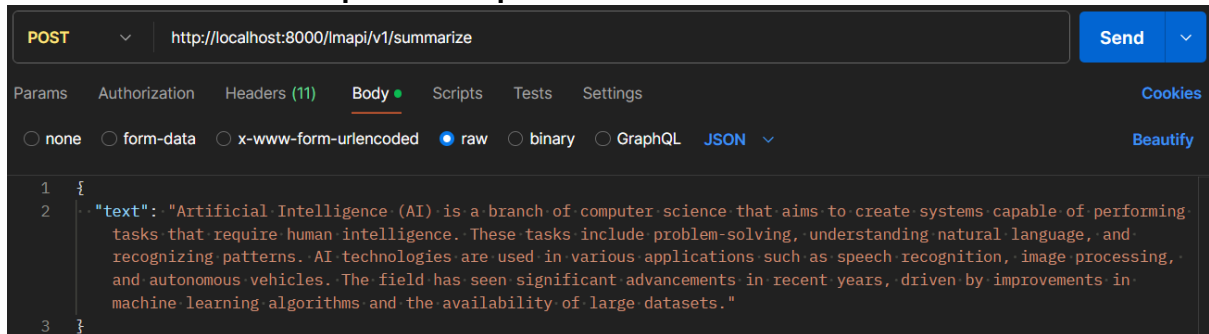
A]. API Setup

This guide covers detailed instructions for configuring and using the API endpoints.
API Endpoints. Demonstrated using Postman Tool.

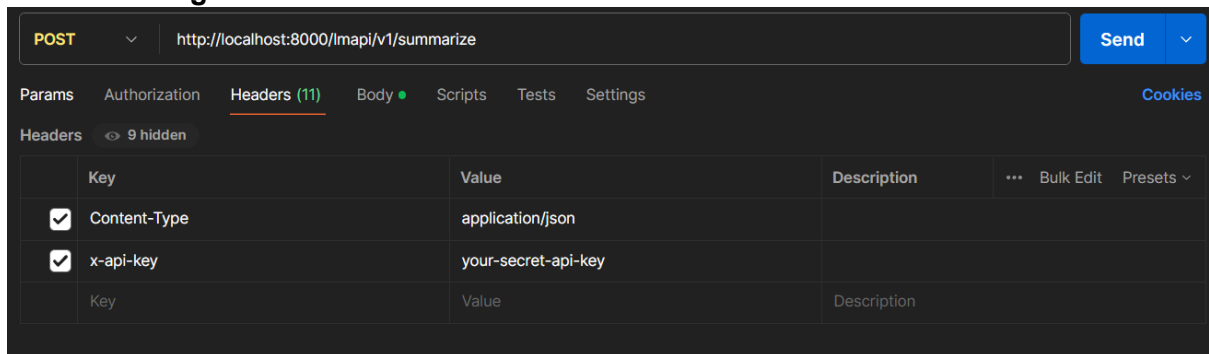
1. Summarize Text

- Endpoint: `/lmapi/v1/summarize`
- Method: POST
- Description: Summarizes a given piece of text into a concise format.

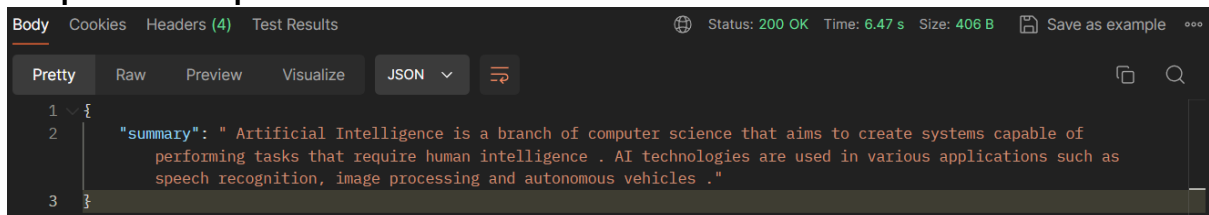
Summarize Text API Request Example:



Header Configuration:



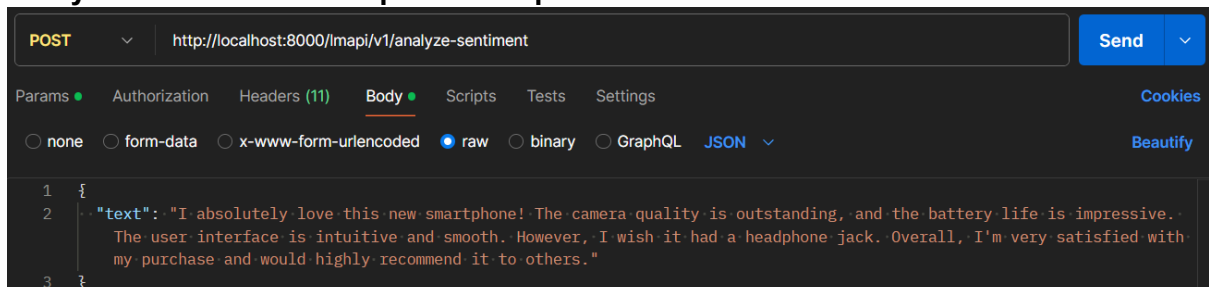
Response Example:



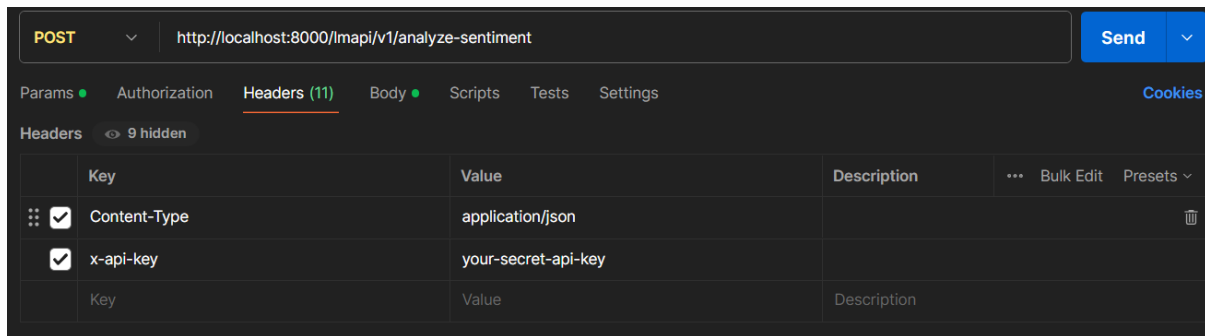
2. Analyze Sentiment

- Endpoint: `/lmapi/v1/analyze-sentiment`
- Method: POST
- Description: Analyzes the sentiment of the given text.

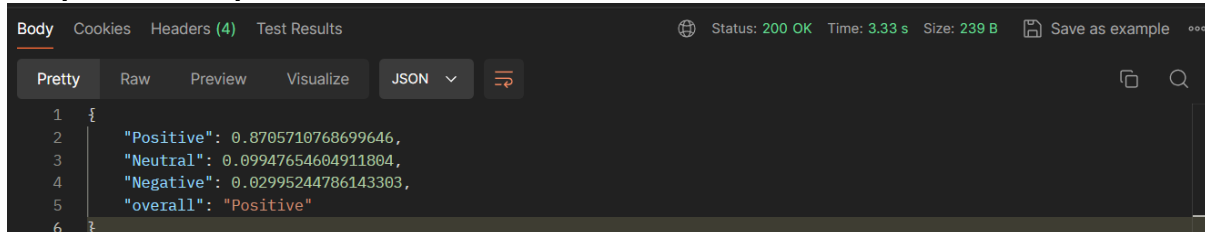
Analyze Sentiment API Request Example:



Header Configuration:



Response Example:



NOTE: More information about API is available at <http://localhost:8000/redoc> when you run the application on localhost.

B]. Configuration

Ensure the `.env` file contains your API key and any other necessary environment variables. Modify `config.py` if you need to adjust settings like `PROJECT_NAME` or `API_V1_STR`.

IV. Design Choices and Considerations

A]. Technologies Selected

- **FastAPI:** Chosen for its asynchronous capabilities, which provide high performance and ease of use. FastAPI automatically generates interactive API documentation, making it easier for developers to understand and test endpoints.
- **React:** Provides a dynamic and responsive user interface with a modern frontend development experience.
- **Hugging Face Transformers:** This tool provides access to state-of-the-art NLP models and allows easy integration with FastAPI. The models are pre-trained, reducing the need for extensive training data and speeding up development.
- **Docker:** Containerization ensures the application can run consistently across different environments, reducing "it works on my machine" issues and simplifying deployment and scaling.
- **Bleach:** It ensures input sanitization and protects the API from cross-site scripting (XSS) and other injection attacks, thus enhancing security.

B]. Quality

- **Use of Established Libraries:** By leveraging established libraries like FastAPI, Transformers, and Pydantic, we ensured high-quality code with minimal bugs and comprehensive type checking.
- **Testing:** The API was thoroughly tested using both automated tests and manual testing through the Swagger UI.

C]. Scalability and Performance

- **Scalability:** The architecture is designed to handle scaling both vertically (increasing resources) and horizontally (adding more instances). Containerisation with Docker supports easy scaling.
- **Model Optimization:** Switched from **facebook/bart-large-cnn** to **sshleifer/distilbart-cnn-12-6**, reducing the response time for summarisation requests from 16-17 seconds to around 8-9 seconds without significantly compromising the quality of the summaries.
- **Asynchronous Processing:** FastAPI's async support allows the API to handle multiple requests simultaneously, improving throughput and performance.

D]. Security

- **API Key Authentication:** Protects the API from unauthorised access by requiring a valid API key for all endpoints.
- **Input Sanitization:** Using bleach to clean all input text prevents common security vulnerabilities such as XSS and injection attacks.

E]. Reliability

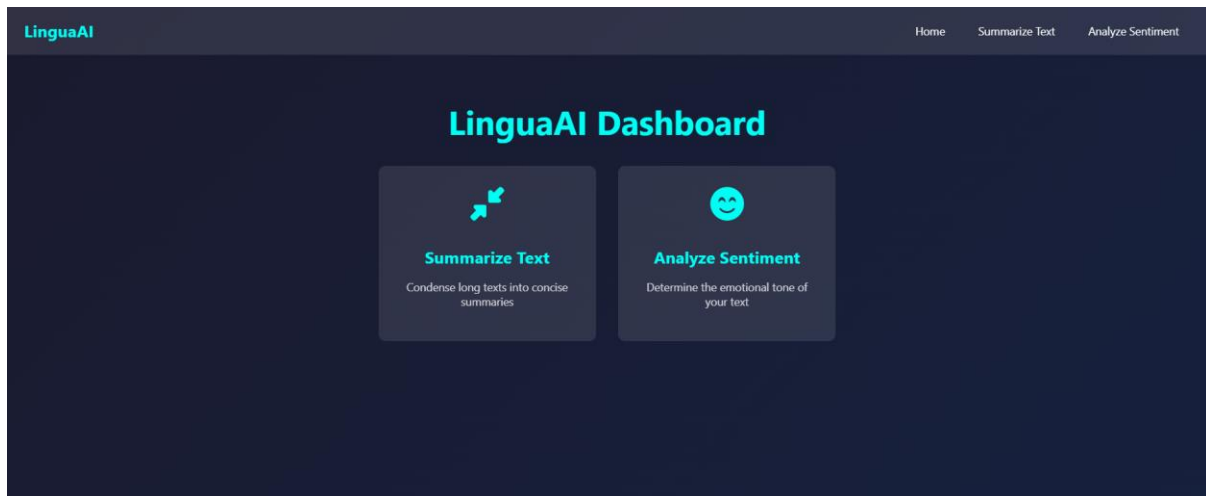
- **Error Handling:** Extensive use of try-except blocks ensures that errors are caught and handled gracefully, providing informative responses to clients and preventing server crashes.
- **Health Checks:** Implementing health check endpoints can be useful for monitoring the service's status in production.

F]. Future Scalability and Maintenance

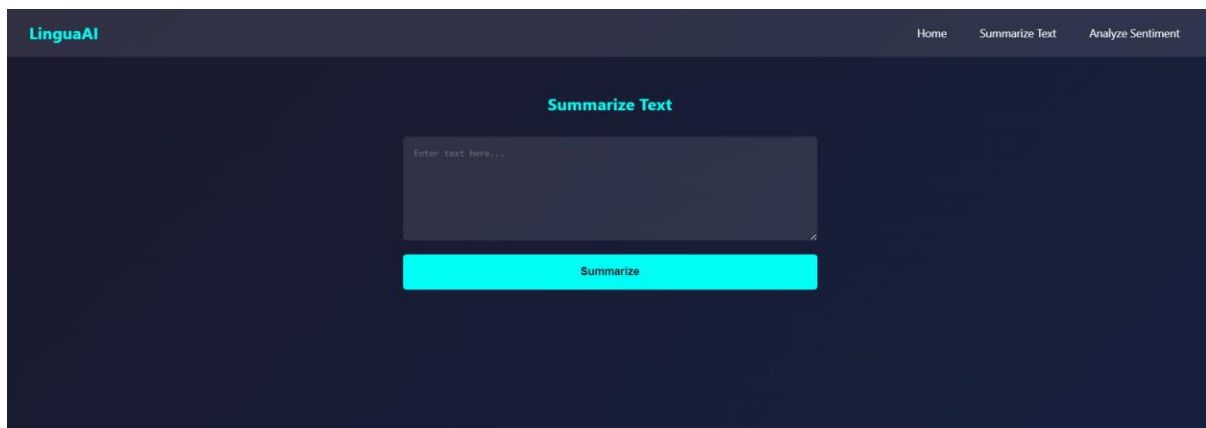
- **Microservices Architecture:** The API design can easily be extended to a microservices architecture, allowing different NLP functionalities to be scaled independently based on demand.
- **Modular Code Structure:** A service layer (LanguageModelService) ensures that logic is decoupled from the API routes, making the codebase easier to maintain and extend.
- **Dockerization:** Simplifies deployment and scaling in cloud environments like AWS, GCP, or Azure, where containers can be orchestrated using Kubernetes or similar tools.
- **Caching:** Implementing caching strategies if performance becomes an issue with increased traffic.

V. User Interface

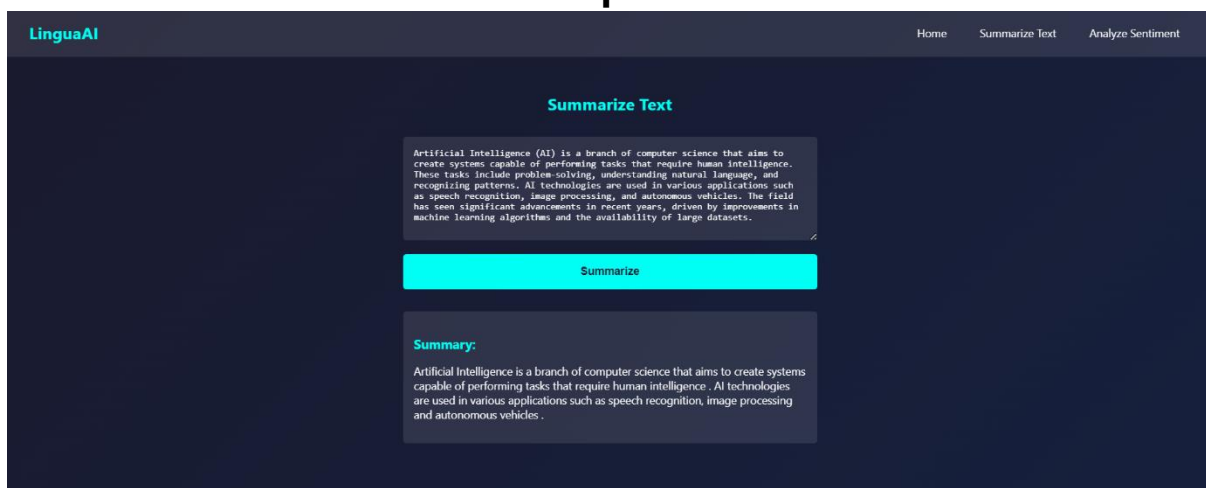
▪ Dashboard



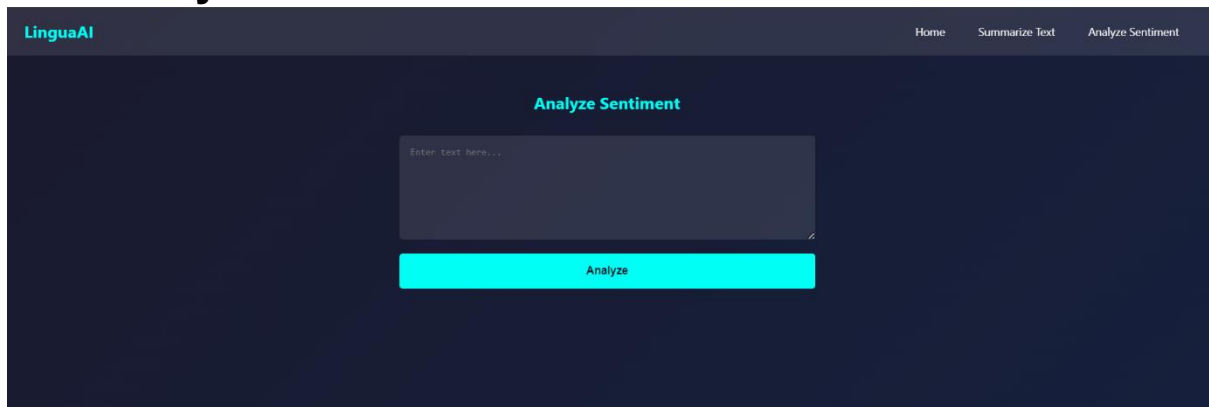
▪ Summarise Text API



▪ Summarise Text API Response



■ Analyse Sentiment API



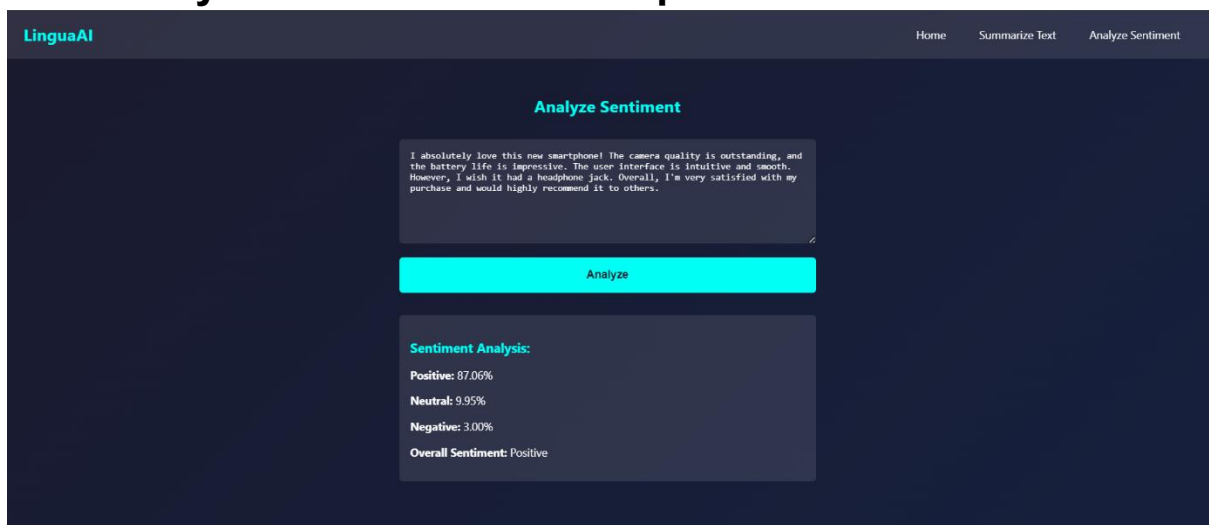
LinguaAI Home Summarize Text Analyze Sentiment

Analyze Sentiment

Enter text here...

Analyze

■ Analyse Sentiment API Response



LinguaAI Home Summarize Text Analyze Sentiment

Analyze Sentiment

I absolutely love this new smartphone! The camera quality is outstanding, and the battery life is impressive. The user interface is intuitive and smooth. However, I wish it had a headphone jack. Overall, I'm very satisfied with my purchase and would highly recommend it to others.

Analyze

Sentiment Analysis:

Positive: 87.06%

Neutral: 9.95%

Negative: 3.00%

Overall Sentiment: Positive

Thank You!