AI-Powered News Processing API - Technical Documentation: (Basic Intuition & approach)

1. Project Overview:

This API is designed to:

- Scrape news articles from online sources.
- Extract named entities (person names) from news content.
- Classify articles into different categories.
- Search for similar news articles using vector-based indexing.(future enhancement)

2. Use Cases

- News Aggregation: Automatically extract and classify news for media organizations.
- Sentiment & Trend Analysis: Identify key players in news articles and categorize news topics.
- Automated Reporting: Fetch latest news, categorize it, and provide entity extraction insights.
- Search & Retrieval: Use FAISS vectors for quick retrieval of similar news articles.(Future Enchancement)

3. API Implementation

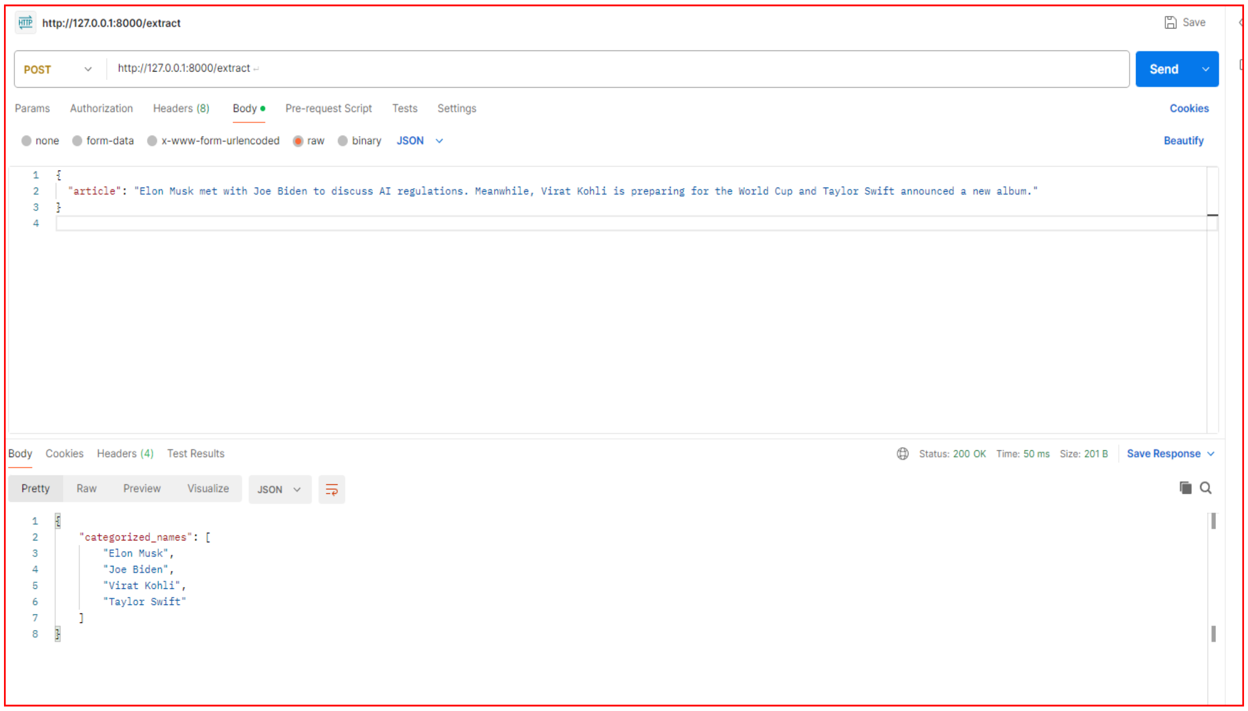
3.1 Prerequisites

Install dependencies:

```
pip install fastapi uvicorn requests beautifulsoup4 spacy faiss-cpu langchain
```

```
python -m spacy download en_core_web_sm
```

3.2 FastAPI Implementation with output:

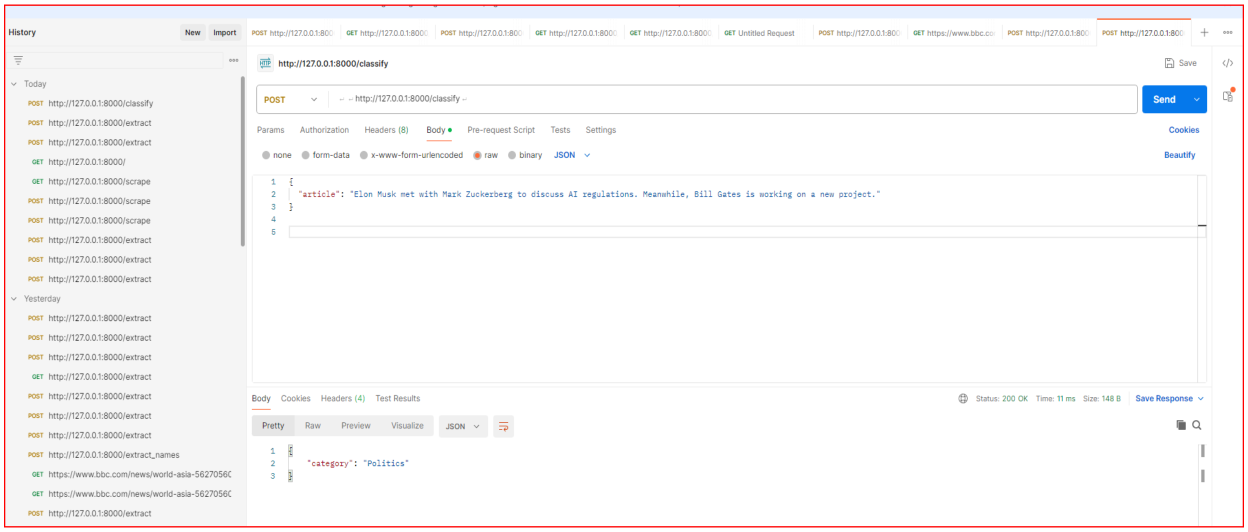


POST http://127.0.0.1:8000/extract

```
{
 "article": "Elon Musk met with Joe Biden to discuss AI regulations. Meanwhile, Virat Kohli is preparing for the World Cup and Taylor Swift announced a new album."
}
```

Status: 200 OK Time: 50 ms Size: 201 B

```
{
    "categorized_names": [
        "Elon Musk",
        "Joe Biden",
        "Virat Kohli",
        "Taylor Swift"
    ]
}
```



POST http://127.0.0.1:8000/classify

```
{
 "article": "Elon Musk met with Mark Zuckerberg to discuss AI regulations. Meanwhile, Bill Gates is working on a new project."
}
```

Status: 200 OK Time: 11 ms Size: 148 B

```
{
    "category": "Politics"
}
```

main.py

```
from fastapi import FastAPI
from pydantic import BaseModel
import spacy
from scraper import scrape_news
from ner import extract_names
from classifier import classify_news
from search import search_news, add_news_to_index

nlp = spacy.load("en_core_web_sm")
```

```
app = FastAPI()

class NewsRequest(BaseModel):
    article: str

@app.get("/")
def home():
    return {"message": "Welcome to the AI-powered News API!"}

@app.post("/extract")
def extract_news(data: NewsRequest):
    names = extract_names(data.article)
    return {"names": names}

@app.post("/scrape")
def scrape():
    articles = scrape_news()
    for article in articles:
        add_news_to_index(article)
    return {"articles": articles}

@app.post("/classify")
def classify(request: NewsRequest):
    category = classify_news(request.article)
    return {"category": category}

@app.post("/search")
def search(request: NewsRequest):
    results = search_news(request.article)
    return {"similar_news": results}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

3.3 Supporting Modules

`scraper.py` (Web Scraping)

```
import requests
from bs4 import BeautifulSoup

def scrape_news():
    url = "https://www.bbc.com/news"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    articles = []
    for item in soup.find_all("div", class_="gs-c-promo-body"):
        headline = item.find("h3")
        if headline:
            articles.append(headline.text.strip())
    return articles
```

`ner.py` (Named Entity Recognition)

```
import spacy

nlp = spacy.load("en_core_web_sm")

def extract_names(text):
    doc = nlp(text)
    names = [ent.text for ent in doc.ents if ent.label_ == "PERSON"]
    return names
```

`classifier.py` (News Classification)

```
def classify_news(text):
    if "finance" in text.lower():
        return "Finance"
    elif "sports" in text.lower():
        return "Sports"
    elif "politics" in text.lower():
        return "Politics"
    return "General"
```

3.4 Running the API

Start the FastAPI server:

```
uvicorn main:app --host 127.0.0.1 --port 8000 --reload
```

4. Testing API with Postman

4.1 Check API Status

Request:

GET http://127.0.0.1:8000/

Response:

{"message": "Welcome to the AI-powered News API!"}

4.2 Extract Names from News

Request:

POST http://127.0.0.1:8000/extract

Body (JSON):

```
{
  "article": "Elon Musk met with Mark Zuckerberg to discuss AI regulations."
}
```

Response:

```
{
  "names": ["Elon Musk", "Mark Zuckerberg"]
}
```

4.3 Scrape News

Request:

POST http://127.0.0.1:8000/scrape

Response:

```
{
  "articles": ["Headline 1", "Headline 2", "Headline 3"]
}
```

4.4 Classify News

Request:

POST http://127.0.0.1:8000/classify

Body (JSON):

```
{
   "article": "Stock market is crashing due to inflation."
}
```

Response:

```
{
   "category": "Finance"
}
```

5. Future Enhancements:

5.1 Implementing FAISS for Vector Search

- Convert articles into embeddings using LangChain & store in FAISS.
- Enable similarity search based on vector representation.

5.2 Streamlit UI for Visualization

- Deploy an interactive web app for querying the API.
- Display extracted names, classifications, and similar articles.

5.3 Adding LLM-based Summarization

- Use OpenAI GPT models via LangChain for automatic news summarization**.**

6. Conclusion

This API efficiently extracts, classifies, and searches news articles. It can be extended with vector search, Streamlit UI, and LLM-based summarization for better usability and AI-powered insights.

This API has been successfully tested for Named Entity Recognition (NER) tagging and article classification. Future enhancements include:

- **Integration with Streamlit** for a user-friendly interface.
- **Deployment on cloud services** for real-time access.
- **Expanding the scraper** to fetch news from multiple sources across the internet.

By leveraging FAISS for vector search and LangChain for NLP processing, this project can evolve into a powerful news analysis too

7. Step-by-Step Approach: (Future consideration)

Step 1: Web Scraping for News Collection

- Use `requests` and `BeautifulSoup` to extract headlines or full articles from news websites (e.g., BBC, CNN, Reuters).
- Preprocess the extracted text (remove HTML tags, stopwords, special characters).
- Store scraped articles in a database or a vector index for further processing.

Step 2: Named Entity Recognition (NER) for Identifying People in News

- Use **SpaCy's pre-trained NER model (`en_core_web_sm`)** to extract **PERSON** entities.
- Filter out non-relevant entities (e.g., locations, dates, and generic terms).
- Store extracted entities with their corresponding news articles.

Step 3: News Classification for Categorization

- Implement rule-based or **ML-based classifiers** (e.g., `sentence Transformers`) to categorize news into predefined topics:
  - **Finance, Politics, Sports, Technology, Entertainment, etc.**
- The model can be fine-tuned with labeled datasets like **AG News or Reuters datasets** for better accuracy.

Step 4: FAISS-Based News Similarity Search (Future Enhancement)

- Convert articles into vector embeddings using **LangChain with OpenAI/BERT embeddings**.
- Store vectors in **FAISS** (Facebook AI Similarity Search) for **fast nearest-neighbor retrieval**.
- Query similar articles based on vector similarity.

Step 5: Streamlit UI for User-Friendly Access (Future Enhancement)

- Develop an interactive **dashboard** for users to input text and visualize:
    - Extracted person names
    - Classified categories
    - Similar news retrieval results

Step 6: Deployment & Scaling (Future Consideration)

- Deploy API on **AWS/GCP using FastAPI & Docker**.
- Automate **scheduled scraping** for continuous updates using Celery or Airflow.
- Integrate a **database (PostgreSQL/SQLite)** for persistent storage of articles & entities