Setup Instructions

Prerequisites:

Python 3.11+

API Keys for OpenAI and Serper (search engine tool)

Installation:

Clone the repo and navigate to project directory.

Create a Python virtual environment and activate it.

Install dependencies from requirements.txt:

bash

pip install -r requirements.txt

Create .env file with:

text

OPENAI_API_KEY=your_openai_api_key_here

SERPER_API_KEY=your_serper_api_key_here

Create a data/ folder and place sample PDF documents inside.

Running the Application:

bash

python main.py

The API will be running at http://localhost:8000.

Usage Instructions

API Endpoints:

Health Check

Method: GET

Path: /

Description: Returns API status.

Document Analysis

Method: POST

Path: /analyze

Parameters:

file: PDF financial document (required).

query: Analysis query string (optional).

Example:

bash

curl -X POST "http://localhost:8000/analyze"

-F "file=@data/sample.pdf"

-F "query=Analyze Tesla's Q2 2025 financials"

API Documentation

Base URL:

http://localhost:8000

Endpoints:

GET /

Health check endpoint to confirm the API is running.

POST /analyze

Upload a financial PDF and receive AI-generated financial analysis.

Request Example:

bash

curl -X POST "http://localhost:8000/analyze"

-F "file=@/path/to/file.pdf"

-F "query=Investment analysis for Q2 2025"

Response:

json

{

"status": "success",

"query": "Investment analysis for Q2 2025",

"analysis": "<AI generated text>",

"file_processed": "file.pdf"

}

Add some more detailed information with these

Here are expanded, detailed, and human-friendly **Setup Instructions, Usage Instructions, and API Documentation** that enhance what you provided:

Setup Instructions

Prerequisites

- You need Python 3.11 or above installed on your computer.
- Obtain API keys for:
  - **OpenAI**: For AI language model integration.
  - **Serper**: For web search functionality enhancing AI responses.

Installation Steps

1. Clone or download the project code to your local machine.
2. Create and activate a Python virtual environment (recommended for clean dependency management):

- For Windows PowerShell:

```
python -m venv env
.\env\Scripts\Activate.ps1
```

- For macOS/Linux Terminal:

```
python -m venv env
source env/bin/activate
```

3. Install required packages (dependencies) with:
   ```
   pip install -r requirements.txt
   ```
4. Create a `.env` file in the project root directory, and add the following keys, replacing placeholders with your own API keys:
   ```
   OPENAI_API_KEY=your_openai_api_key_here
   SERPER_API_KEY=your_serper_api_key_here
   ```
5. Create a directory named `data/` in the project root. This folder will hold your financial PDF files to be analyzed.

Usage Instructions

Running the Application

To start the FastAPI server, execute:

```
python main.py
```

This will launch the API server accessible by default at:

```
http://localhost:8000
```

You can open this URL in your browser to verify that the server is running.

How to Use the API

Health Check Endpoint

- **HTTP Method**: GET
- **URL**: `/`
- **Description**: Verify that the server is running and ready to accept requests.

Document Analysis Endpoint

- **HTTP Method**: POST
- **URL**: `/analyze`
- **Parameters**:
  - `file` (required): The financial report PDF file to analyze.
  - `query` (optional): A natural language query or instruction for the AI analysis. Defaults to analyzing the document for investment insights.

Example using `curl`:

```
curl -X POST "http://localhost:8000/analyze" \
  -F "file=@data/sample.pdf" \
  -F "query=Analyze Tesla's Q2 2025 financial performance"
```

Example using Python's `requests` module:

```
import requests

with open('data/sample.pdf', 'rb') as f:
    files = {'file': f}
    data = {'query': 'What are the key revenue trends and risks?'}
    response = requests.post('http://localhost:8000/analyze', files=files, data=data)
    print(response.json())
```

API Documentation for Financial Document Analyzer

Base URL

```
http://localhost:8000
```

Available Endpoints

1. Health Check

- **Method:** GET
- **URL:** `/`
- **Description:** Simple endpoint to check if the API server is alive and functioning.

- **Response:**
  - Status code: `200 OK`
  - Body (JSON):

```
{
  "message": "Financial Document Analyzer API is running"
}
```

2. Analyze Financial Document

- **Method:** POST
- **URL:** `/analyze`
- **Description:** Analyse an uploaded financial PDF document based on a query and return AI-generated insights.
- **Parameters:**
  - `file` (form-data, required): PDF document to analyze.
  - `query` (form-data, optional): Textual query or instructions for analysis (default: `"Analyze this financial document for investment insights"`).
- **Request Example:**
  Use `curl` to upload and query:

```
curl -X POST "http://localhost:8000/analyze" \
  -F "file=@/path/to/financial_report.pdf" \
  -F "query=Investment analysis for Q2 2025"
```

- **Response Example:**
  - Status code: `200 OK`
  - Body (JSON):

```
{
  "status": "success",
  "query": "Investment analysis for Q2 2025",
  "analysis": "Tesla's Q2 2025 revenue grew by X%, showing signs of ...",
  "file_processed": "financial_report.pdf"
}
```

- **Error Handling:**
  - Returns HTTP 400 for unsupported file types.
  - HTTP 500 for processing errors with meaningful error details.

3. Optional: Check Async Task Status

(If you implement Celery for async processing)

- **Method:** GET
- **URL:** `/status/{task_id}`
- **Description:** Returns the processing status and result of document analysis tasks.
- **Response:**
    - Task status (pending, processing, completed, failed)
    - Analysis result when done
- **Usage:** Allows clients to poll results asynchronously.

Response Models and Status Codes

Use FastAPI's `response_model` feature to define clear response schemas, improving auto-generated docs and client validation.

- Use standard HTTP status codes:
    - `200 OK` for successful requests
    - `400 Bad Request` for invalid inputs
    - `500 Internal Server Error` for unexpected failures

Documentation Usability Features

- **Interactive Docs:** Navigate to http://localhost:8000/docs or http://localhost:8000/redoc to explore the interactive OpenAPI documentation automatically generated by FastAPI.
- **Endpoint Tags:** Organize endpoints by tags such as `Health` and `Financial Analysis` for clarity.
- **Descriptions and Summaries:** Each endpoint includes human-readable summaries and parameter explanations for ease of use.
- **Examples:** Use complete request and response examples to guide API users.

Best Practices

- **Clear Parameter Validation:** Use Pydantic models to validate incoming requests and serialize responses.
- **Handle large file uploads gracefully:** Check file size and type on upload to avoid server overload or crashes.
- **Secure your API:** Use HTTPS, API keys, or OAuth for production environments.
- **Limit request rates:** Prevent abuse by setting rate limits per client IP or API key.
- **Log and monitor requests:** Enable structured logging and health monitoring for operational visibility.