**📘 API Documentation for RAG Resume App**

This document provides a comprehensive overview of the **APIs used in the RAG Resume Application**. The document covers all the **Flask API endpoints**, **external APIs** (like OpenAI's API), and **internal functions** used within the app. It serves as a reference for future development, debugging, and maintenance.

**🔥 Table of Contents**

1. **Flask API Endpoints**
   * [1. / (GET)](#1-root-get)
   * [2. /ask (POST)](#2-ask-post)
2. **External API Usage**
   * [1. OpenAI API](#openai-api)
3. **Internal Functions**
   * [1. initialize\_rag\_system()](#initialize_rag_system)
   * [2. process\_text()](#process_text)
   * [3. create\_qa\_chain()](#create_qa_chain)
   * [4. extract\_text\_from\_pdf()](#extract_text_from_pdf)

**📍 Flask API Endpoints**

**1️⃣ / (GET)**

**URL**

arduino

Copy code

http://127.0.0.1:5002/

**Method**

sql

Copy code

GET

**Description**

* This endpoint serves the **index.html** file, which is the user interface for the RAG application.
* This page allows users to ask questions about the resume and optionally upload new files.

**Response**

* **200 OK** — Returns the HTML page successfully.
* **500 Internal Server Error** — If there is an error loading the page.

**Example Request**

arduino

Copy code

GET http://127.0.0.1:5002/

**Response Example**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>RAG Resume App</title>

</head>

<body>

<h1>Welcome to RAG Resume App</h1>

</body>

</html>

**2️⃣ /ask (POST)**

**URL**

arduino

Copy code

http://127.0.0.1:5002/ask

**Method**

Copy code

POST

**Description**

* This endpoint allows users to send a **question** related to the resume.
* The app processes the question using the **QA chain** and returns a response from the **GPT-4-turbo** model.

**Request Body**

| **Field** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| **question** | **string** | **Yes** | The user's question about the resume. |

**Example Request**

json

Copy code

{

"question": "What is the applicant's work experience?"

}

**Response**

* **200 OK** — Returns the AI-generated response to the user's question.
* **400 Bad Request** — If the question is missing from the request.
* **500 Internal Server Error** — If the QA chain is not initialized or an internal error occurs.

**Response Example**

**Success Response (200)**

json

Copy code

{

"response": "The applicant has 5 years of experience in software development."

}

**Error Response (400)**

json

Copy code

{

"error": "No question received"

}

**Error Response (500)**

json

Copy code

{

"error": "QA chain is not initialized"

}

**🌐 External API Usage**

**🔥 OpenAI API**

This app uses the **OpenAI Embeddings API** and **GPT-4-turbo** for natural language processing and question answering.

**API: POST /v1/embeddings**

**Base URL**:

bash

Copy code

https://api.openai.com/v1/embeddings

**Request Headers**

| **Header** | **Value** |
| --- | --- |
| Authorization | Bearer <OPENAI\_API\_KEY> |
| Content-Type | application/json |

**Request Body**

json

Copy code

{

"input": "This is a sample text for embedding.",

"model": "text-embedding-ada-002"

}

**Response Example**

json

Copy code

{

"object": "embedding",

"data": [

{

"embedding": [0.1, 0.2, 0.3, ...],

"index": 0

}

],

"model": "text-embedding-ada-002"

}

**API: POST /v1/chat/completions**

**Base URL**:

bash

Copy code

https://api.openai.com/v1/chat/completions

**Request Headers**

| **Header** | **Value** |
| --- | --- |
| Authorization | Bearer <OPENAI\_API\_KEY> |
| Content-Type | application/json |

**Request Body**

json

Copy code

{

"model": "gpt-4-turbo",

"messages": [{"role": "user", "content": "What is the work experience of this person?"}],

"temperature": 0

}

**Response Example**

json

Copy code

{

"choices": [

{

"message": {

"role": "assistant",

"content": "The applicant has 5 years of experience in software development."

}

}

]

}

**⚙️ Internal Functions**

**1️⃣ initialize\_rag\_system()**

* **Location**: **src/main.py**
* **Description**:
  + Loads the PDF from the **/data** folder.
  + Extracts text from the PDF.
  + Creates embeddings, vector store, and QA chain.

**2️⃣ process\_text(text)**

* **Location**: **src/rag.py**
* **Input**:
  + **text** (string): The text extracted from the PDF.
* **Output**:
  + A **vector store** containing embeddings of the text.

**3️⃣ create\_qa\_chain(vector\_store)**

* **Location**: **src/rag.py**
* **Input**:
  + **vector\_store**: The FAISS vector store created from the processed text.
* **Output**:
  + A **QA chain** that enables Q&A using GPT-4-turbo.

**4️⃣ extract\_text\_from\_pdf(file\_path)**

* **Location**: **src/utils.py**
* **Input**:
  + **file\_path** (string): Path to the PDF file.
* **Output**:
  + **Text** (string): The extracted text from the PDF.
* **Description**:
  + Uses **PyPDF2** to extract text from the PDF.
  + If the PDF is image-based, this function will fail.

**📋 Summary**

| **API** | **Method** | **Path** | **Description** |
| --- | --- | --- | --- |
| **Root** | **GET** | / | Serves index.html for the UI. |
| **Ask** | **POST** | /ask | Asks a question to the RAG system. |
| **OpenAI API** | **POST** | /v1/embeddings | Embeds text into vector. |
| **OpenAI API** | **POST** | /v1/chat/completions | Uses GPT-4-turbo for Q&A. |

**🛠️ How to Test the APIs**

1. **Run the Flask app**:

bash

Copy code

python src/main.py

1. **Test the /ask API** using cURL:

bash

Copy code

curl -X POST http://127.0.0.1:5002/ask -H "Content-Type: application/json" -d '{"question": "What is the applicant\'s experience?"}'

1. **Test OpenAI API** directly (if needed):

bash

Copy code

curl https://api.openai.com/v1/models \

-H "Authorization: Bearer <OPENAI\_API\_KEY>"